The optimal design of low-latency virtual backbones

Hamidreza Validi and Austin Buchanan

School of Industrial Engineering & Management
Oklahoma State University
{hamidreza.validi,buchanan}@okstate.edu

June 3, 2019

Abstract

Two nodes of a wireless network may not be able to communicate with each other directly perhaps due to obstacles or insufficient signal strength. This necessitates the use of intermediate nodes to relay information. Often, one designates a (preferably small) subset of them to relay these messages (i.e., to serve as a virtual backbone for the wireless network) which can be seen as a connected dominating set (CDS) of the associated graph. Ideally, these communication paths should be short, leading to the notion of a latency-constrained CDS. In this paper, we point out several shortcomings of a previously studied formalization of a latency-constrained CDS and propose an alternative one. We introduce an integer programming formulation for the problem that has a variable for each node and imposes the latency constraints via an exponential number of cut-like inequalities. Two nice properties of this formulation are that: (1) it applies when distances are hop-based and also when they are weighted; and (2) it easily generalizes to ensure fault tolerance. We provide a branch-and-cut implementation of this formulation and compare it with a new polynomial-size formulation. Computational experiments demonstrate the superiority of the cut-like formulation. We also study related questions from computational complexity such as approximation hardness and answer an open problem regarding the fault diameter of graphs.

Keywords: connected dominating set; strongly connected dominating set; latency; delay; hop constraint; k-club; length-bounded cut; wireless networks; fault-tolerant; integer programming;

1 Introduction

Two nodes of a wireless network may not be able to communicate with each other *directly* perhaps due to obstacles or insufficient signal strength. This necessitates the use of intermediate nodes to relay information. Often, one designates a small subset of them to relay messages (i.e., to serve as a virtual backbone for the wireless network) which amounts to a connected dominating set of the associated graph, defined below.

Definition 1 (CDS)**.** *A subset $D \subseteq V$ of vertices is a connected dominating set (CDS) for an undirected graph $G = (V, E)$ if:*

1. *D is* dominating*, i.e., every vertex from $V \setminus D$ neighbors a vertex of D; and*
2. *D is* connected*, i.e., the subgraph $G[D]$ induced by D is connected.*

If the graph G is not complete[1], a CDS can equivalently be defined as a subset $D \subseteq V$ of vertices such that, for every vertex pair $\{a, b\} \in \binom{V}{2}$, there exists a path connecting a and b whose interior vertices belong to D.

A CDS ensures that the nodes of the network can communicate with each other. It provides little guarantee on *how long* it will take for a message to be received once it has been sent. This has led some researchers to impose additional constraints on the CDS $D \subseteq V$, namely that the subgraph induced by the dominating set D has diameter at most s, i.e., is a dominating s-club (Li et al., 2007; Zhang et al., 2008; Buchanan et al., 2014). This ensures that messages will be received in $s + 2$ hops: one hop to reach the CDS, at most s hops within the CDS, and one hop to reach the destination.

Definition 2 (Dominating s-club)**.** *A subset $D \subseteq V$ of vertices is a dominating s-club for an undirected graph $G = (V, E)$ if:*

1. *D is* dominating*, i.e., every vertex from $V \setminus D$ neighbors a vertex of D; and*
2. *D is an s-*club*, i.e., the subgraph $G[D]$ induced by D has diameter at most s.*

We argue that this formalization of the problem is less than ideal. First, and most importantly, a dominating s-club does not quite capture the intent of the hop constraints, as we will illustrate. Suppose that we want a CDS that facilitates 4-hop communication in the graph in Figure 1. This can be ensured by the dominating 2-club given in Figure 1(a). Indeed, a message sent from node 5 to node 8 through this virtual backbone must follow the path 5-4-3-6-8, which takes four hops.

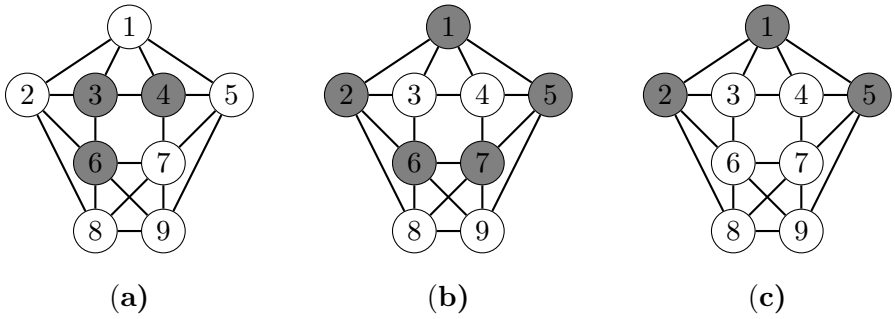


Figure 1: (**a**) dominating 2-club; (**b**) latency-2 CDS; (**c**) latency-3 CDS.

If 3-hop communication were required, one might search for a dominating 1-club, but none exist in this graph. This may lead us to believe that a CDS that facilitates 3-hop communication does not exist, but this belief would be false. Indeed, in Figure 1(b) we provide a CDS which needs at most *two* hops to transmit information, so we could call it a latency-2 CDS. Further, Figure 1(c) gives a latency-3 CDS which is also a minimum CDS! Note that a message can be passed *directly* from node 8 to node 9 in the wireless network since they are adjacent; it does *not* have to be relayed through the CDS nodes.

Another limitation of previous works is that they make the simplifying assumption that distances are measured by the number of hops, see Li et al. (2007); Zhang et al. (2008); Buchanan et al. (2014) and Chapter 7 of Du and Wan (2013). However, this may ultimately provide a poor approximation to the actual end-to-end delay when the delays at the nodes differ. For example, a particular

[1]The complete graph is the only exception. In this case, no virtual backbone is needed and yet Definition 1 would disallow the empty set. For this reason, the complete graph is treated "with generous disregard" in the virtual backbone literature.

node may play a central role in the CDS, needing to relay a large number of messages. This may cause messages to have to wait to be transmitted, and these *queueing delays* may be more realistically captured for our purposes via *node-weighted* delays as opposed to hop-based delays. For more information about this and other delays in wireless networks, consult Xie and Haenggi (2009) and Zhong et al. (2017).

With these shortcomings in mind, we propose a new formalization of a low-latency virtual backbone, which we call a latency-s CDS. For purposes of generality, it is defined in terms of a *directed* and—without loss of generality—*edge-weighted* graph. By allowing for directed edges, we can model non-uniform transmission ranges. For example, consider the case where a node i has a large transmission range and is far away from a node j that has a small transmission range. In this scenario, the edge (i, j) should exist, but not the edge (j, i). Note that, as we make the transition to directed graphs, we are no longer referring to a "CDS" in the sense of Definition 1, but rather in terms of a *strongly connected dominating set* in the sense of Li et al. (2009). This is defined as a subset D of vertices such that: (i) D induces a strongly connected subgraph; and (ii) every vertex from $V \setminus D$ has both an in-neighbor and an out-neighbor in D.

Definition 3 (latency-s CDS)**.** *A vertex subset $D \subseteq V$ is a latency-s CDS for a directed graph $G = (V, E)$ under edge weights $w : E \to \mathbb{R}_+$ if, for every vertex pair $(a, b) \in V \times V$, there is a path from a to b of length at most s whose interior vertices belong to D.*

Observe that the graph G in Definition 3 is edge-weighted but not vertex-weighted. This is without loss of generality, as the following will illustrate. Consider the 3-vertex, undirected path graph 1-2-3 representing a wireless network. Sending a message from node 1 to node 3 would incur delays at nodes 1 and 2 (since the delay is based on the transmitting node), as well as delays on edges $\{1, 2\}$ and $\{2, 3\}$, for an end-to-end delay that might be denoted $d_1 + d_2 + d_{\{1,2\}} + d_{\{2,3\}}$. Instead, we can replace each undirected edge $\{i, j\}$ by its directed counterparts (i, j) and (j, i) and let the delay of each directed edge $d_{(i,j)}$ be the delay of its undirected counterpart $d_{\{i,j\}}$ plus the delay of its tail node d_i. In this way, it is sufficient to consider a directed graph with only edge weights.

Definition 3 overcomes the aforementioned issues with the previous formalization based on dominating s-clubs. As an added bonus, it has superior computational properties. Indeed, checking whether a graph admits a latency-s CDS is as simple as checking whether the graph's diameter is at most s. In contrast, the problem of checking whether there exists *any* dominating s-club is NP-complete; specifically, this is true under hop-based distances for the two most restrictive (but nontrivial) cases where $s = \text{diam}(G) - 2$ (Schaudt, 2013) and $s = \text{diam}(G) - 1$ (Buchanan et al., 2014), where diam denotes the graph's diameter.

The associated optimization problem is as follows.

Problem: The minimum latency-s CDS problem.
Input: A directed graph $G = (V, E)$, a weight $w_e \geq 0$ for each edge $e \in E$, and a number s.
Output: (if any exist) A smallest subset $D \subseteq V$ of vertices that is a latency-s CDS.

In this paper, we propose an integer programming (IP) formulation for this problem that uses an exponential number of cut-like inequalities. As we will see, a relatively simple implementation of it significantly outperforms a polynomial-size formulation that we introduce. This second formulation has $O(sn^2)$ variables and $O(snm)$ constraints and applies when the distances are hop-based, where n and m denote the number of vertices and edges, respectively. In contrast, the cut-like formulation applies when there are weighted delays.

1.1 Previous Work

The minimum CDS problem is a well-studied NP-hard problem (Garey and Johnson, 1979) in which the task is to find a CDS of minimum cardinality. For example, Figure 2(a) provides a CDS and Figure 2(b) provides a *minimum* CDS. The reader is encouraged to consult the book by Du and Wan (2013) for motivating applications, approximation algorithms, and hardness results. There are a number of IP formulations and implementations for the minimum CDS problem and for the equivalent maximum-leaf spanning tree problem (Lucena et al., 2010; Simonetti et al., 2011; Morgan and Grout, 2008; Fan and Watson, 2012; Fujie, 2004; Gendron et al., 2014; Buchanan et al., 2015). See also the literature on the regenerator location problem (Chen et al., 2010, 2015; Li and Aneja, 2017). To our knowledge, the state-of-the-art[2] IP formulation and implementation are due to Fujie (2004) and Buchanan et al. (2015), respectively, although several of the previously mentioned approaches work well. As far as we know, the only previous work to propose an IP formulation for a latency-constrained variant of the CDS problem is by Buchanan et al. (2014); however, it is for dominating s-clubs.

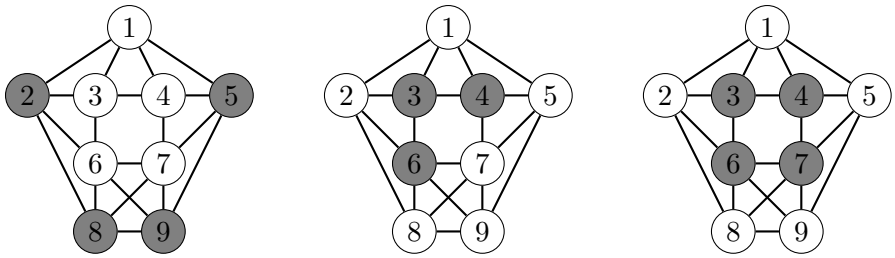


Figure 2: (**a**) CDS; (**b**) *minimum* CDS; (**c**) 2-connected 2-dominating set.

Assuming the input graph is not complete, the minimum CDS problem can be formulated as an IP as follows, where x_i is a binary variable representing the decision to include vertex i in the CDS. A vertex cut is a subset $C \subset V$ of vertices such that $G - C := G[V \setminus C]$ is disconnected and nontrivial (i.e., has at least two nodes).

$$\min \sum_{i \in V} x_i \tag{1}$$

$$\sum_{i \in C} x_i \geq 1, \ \forall \text{ vertex cut } C \subset V \tag{2}$$

$$x_i \in \{0, 1\}, \ \forall i \in V. \tag{3}$$

This particularly elegant formulation is essentially due to Fujie (2004), and its linear programming relaxation can be solved in polynomial time despite having exponentially many constraints, as the separation problem for the vertex cut constraints (2) can be solved in polynomial time. The implementation of Buchanan et al. (2015) used this formulation to solve 42 of 47 standard test

[2] An Associate Editor referred us to a later paper by Li and Aneja (2017) that proposes to use the same Fujie-based formulation but with some additional cuts. Li and Aneja claim that their two implementations, named B&C1 and B&C2, "significantly outperformed the available exact algorithm[s] in the literature," but neglect to compare results with Buchanan et al. (2015). This is problematic as Li and Aneja fail to solve 2 of these 47 instances within a time limit of 3600 seconds. Namely, Li and Aneja do not solve the instances `v200_d10` and `IEEE-300-Bus` within the time limit, but Buchanan et al. solve these instances in 496.43 and 52.88 seconds, respectively.

instances each in under 10 seconds (and never taking longer than 500 seconds), whereas no earlier approach solved 42 instances each in a 1-hour time limit. In their implementation, Buchanan et al. (2015) add violated vertex cut inequalities on-the-fly, cutting off infeasible *integer* points. Our proposed formulation in this paper, which generalizes Fujie's formulation, is implemented in the same manner.

One drawback of a CDS is that it can be vulnerable to node or arc failures. For example, consider the minimum CDS from Figure 2(b). If node 3 fails, this renders the virtual backbone inoperative as it no longer can relay information (say, from node 5 to node 8).

This motivates the notion of a fault-tolerant CDS—one that remains a CDS when fewer than k nodes fail. This has been called a k-connected k-dominating set (k-k-CDS) as it can equivalently be defined as a subset $S \subseteq V$ of vertices such that $G[S]$ is k-vertex-connected and every vertex of $V \setminus S$ has k neighbors in S. Figure 2(c) gives a 2-2-CDS, which remains a CDS if one vertex fails. The associated optimization problem, the minimum k-k-CDS problem, admits the following formulation (Buchanan et al., 2015; Ahn and Park, 2015).

$$\min \sum_{i \in V} x_i \tag{4}$$

$$\sum_{i \in C} x_i \geq k, \ \forall \text{ vertex cut } C \subset V \tag{5}$$

$$x_i \in \{0, 1\}, \ \forall i \in V. \tag{6}$$

The formulations that we propose in this paper generalize this k-k-CDS formulation as well as Fujie's CDS formulation.

1.2 Notation and Terminology

From now on, unless stated otherwise, $G = (V, E)$ will be a directed graph, with vertex set V and edge set $E \subset V \times V$, that has no loops and no parallel edges. By "no parallel edges", we mean that there is at most one directed edge from a vertex i to a vertex j, and so we can refer to it by the notation (i, j). Here, i is called the tail and j is the head. Frequently, we *bidirect* an undirected edge, which we define to be the operation in which an undirected edge $\{i, j\}$ is replaced by its directed counterparts (i, j) and (j, i). When the edges of G are reciprocated, i.e., if $(i, j) \in E$ implies $(j, i) \in E$, then we say that G is bidirected—not to be confused with the bidirected graphs of Edmonds and Johnson (1970), see also Schrijver (2003).

For each edge $e \in E$ of G, there is an associated nonnegative weight w_e representing the delay. In the *hop-based* case, each weight is one. The distance from vertex a to vertex b in graph G, denoted $\text{dist}_G(a, b)$, is the length of a shortest path from a to b in G, edge-weighted by w. Convention states that if there is no path from a to b in G, then $\text{dist}_G(a, b) = \infty$. The diameter of G, denoted $\text{diam}(G)$, is the maximum of these pair-wise distances, i.e., $\text{diam}(G) := \max\{\text{dist}_G(a, b) \mid a, b \in V\}$.

The out-neighborhood and in-neighborhood of a vertex $v \in V$ in G are denoted $N_G^+(v) := \{w \in V \mid (v, w) \in E\}$ and $N^-(v) := \{u \in V \mid (u, v) \in E\}$, respectively. For a vertex subset S, $\delta_G^+(S)$ denotes the subset of edges whose tail belongs to S and whose head does not. Similarly, $\delta_G^-(S)$ denotes the subset of edges whose head belongs to S but whose tail does not. For a singleton $S = \{v\}$, let $\delta_G^+(v) := \delta_G^+(\{v\})$ and $\delta_G^-(v) := \delta_G^-(\{v\})$. When the graph G in question is clear, the subscripts G in $N_G^+(\cdot)$, $N_G^-(\cdot)$, $\delta_G^+(\cdot)$, and $\delta_G^-(\cdot)$ are omitted. The subset of edges having both endpoints in $S \subseteq V$ is denoted $E(S) := \{(i, j) \in E \mid i, j \in S\}$.

1.3 Our Contributions

In Section 2, we examine the complexity of latency-s CDS's. Specifically, we answer questions like: How quickly can one verify that a given subset of vertices is a latency-s CDS? And, how hard is the minimum latency-s CDS problem?

In Section 3, we propose IP formulations for the minimum latency-s CDS problem. The first formulation, which we call CUT, has n binary variables and an exponential number of cut-like constraints. We then generalize this formulation so that it models the fault-tolerant variant in which one seeks a latency-s CDS that maintains feasibility after a small number of vertex failures. Then, we give a second IP formulation, which we call POLY, that has $O(sn^2)$ variables and $O(snm)$ constraints. It serves as a baseline for computational comparisons.

In Section 4, we examine the complexity of the separation problem associated with formulation CUT. Specifically, we show that, under hop-based distances, it is polynomial-time solvable for $s \in \{2, 3, 4\}$ and NP-hard when $s \geq 5$. En route to proving this, we answer an open question of Xu et al. (2005), by showing that it is indeed NP-hard to compute a graph's fault diameter.

In Section 5, we perform computational experiments. Our results demonstrate that a branch-and-cut implementation of formulation CUT significantly outperforms the polynomial-size formulation POLY. Notably, CUT makes easy work of a real-life instance with 300 nodes, while formulation POLY struggles to solve instances with 50 nodes in an hour.

In Section 6, we conclude and discuss directions for future research.

2 The Complexity of Latency-s CDS

In this section, we examine the complexity of latency-s CDS's. First, we pinpoint the complexity of verifying feasible solutions, showing essentially that a quadratic running time is unavoidable under a plausible complexity assumption. Then, we establish the inapproximability of the minimum latency-s CDS problem.

2.1 The Complexity of Verifying Feasible Solutions

To verify that a given subset $D \subseteq V$ of vertices is a latency-s CDS, we can compute, for each vertex $v \in V$, the shortest paths from v to all other nodes $t \in V \setminus \{v\}$ and check that these paths are short enough. However, we are not interested in just any paths from v to t; these paths must not cross vertices from $V \setminus D$. In our proposed approach, we solve an instance of the single source shortest path problem (SSSP) in the subgraph $\overrightarrow{G}_v^D = (V, \overrightarrow{E}_v^D)$, which has edge set

$$\overrightarrow{E}_v^D := E(D) \cup \delta^+(D) \cup \left(\delta^+(v) \cap \delta^-(D)\right). \tag{7}$$

Here, we preserve the edges $E(D)$ that have both endpoints in D, those edges $\delta^+(D)$ that point out of D, and those edges $\delta^+(v) \cap \delta^-(D)$ whose tail is v and whose head is in D. This set $\overrightarrow{E}_v^D$ includes all edges that might be used in a suitable path from v to another node. Of course, we need not create the graph $\overrightarrow{G}_v^D$ in the implementation, as nearly any shortest path algorithm can be reconfigured to work implicitly on $\overrightarrow{G}_v^D$ when given G, D, and v.

IsLatencyConstrainedCDS(G, D, s):

1. for each $v \in V$ do
   (a) compute shortest paths from v in $\overrightarrow{G}_v^D$;
   (b) if $\text{dist}_{\overrightarrow{G}_v^D}(v, t) > s$ for some $t \in V \setminus \{v\}$, then return "no";

2. return "yes".

Proposition 1. *The algorithm* IsLatencyConstrainedCDS *correctly determines whether a given subset $D \subseteq V$ of vertices is a latency-s CDS for a directed graph $G = (V, E)$:*

- *in $O(mn + n^2 \log\log n)$ time and linear space under nonnegative edge weights;*
- *in $O(mn)$ time and linear space in the hop-based case.*

Here, we are using the algorithm of Thorup (2004) to compute SSSP in time $O(m + n \log\log n)$ in the nonnegative weights case, and BFS to solve SSSP in the hop-based case.

Given that this or some other verification procedure will be called repeatedly in our implementation, it is important that it runs as quickly as possible. For example, we would like to know: is there a different verification procedure that, say, runs in linear time $O(m + n)$? Unfortunately, under a complexity assumption called the strong exponential time hypothesis (SETH) of Impagliazzo et al. (2001) and Impagliazzo and Paturi (2001), this is not possible.

Proposition 2. *If SETH holds, then for every $\varepsilon > 0$ there exists no algorithm for verifying that a subset D of vertices is a latency-s CDS that runs in time $O(m^{2-\varepsilon})$, even in the simplest nontrivial case of hop-based distances and $s = 2$.*

The proof and discussions regarding the limitations of local search are provided in Section 1 of the Online Supplement.

2.2 The Inapproximability of the Minimum Latency-s CDS Problem

We provide a hardness result for approximating the size of a minimum latency-s CDS. It is based on the hardness result of Dinur and Steurer (2014) which states that approximating the minimum hitting set problem to within a factor of $(1-\varepsilon)\ln h$ is NP-hard for every $\varepsilon > 0$, where h refers to the number of subsets to hit, cf. Raz and Safra (1997); Alon et al. (2006); Moshkovitz (2012). Also, see similar hardness results based on the stronger assumption that NP does not have quasipolynomial-time algorithms (Lund and Yannakakis, 1994; Feige, 1998). Note that $|U| = O(h^c)$ for some constant c in Dinur and Steurer's result.

Problem: The minimum hitting set problem.
Input: a family $F_1, \ldots, F_h \subseteq U$ of subsets of U.
Output: A minimum cardinality subset $D \subseteq U$ such that $|D \cap F_i| \geq 1$ for every $i = 1, \ldots, h$.

Theorem 1. *There exists a polynomial-time algorithm that, when given an instance $((F_1, \ldots, F_h), U)$ of the minimum hitting set problem, creates an instance $(G = (V, E), w, s)$ of the minimum latency-s CDS problem that satisfies:*

- *$|V| = 4 + h + |U|$ and $s = 2 = \mathrm{diam}(G)$ and $w_e = 1$ for each $e \in E$;*
- *there exists a k-hitting set if and only if there exists a $(k+2)$-vertex latency-s CDS.*

Proof. Let $V = \{r, a, b, c\} \cup T \cup U$, where $T = \{t_1, \ldots, t_h\}$. Thus, $|V| = 4 + h + |U|$. Construct E by bidirecting the following edges. Connect r to every vertex of $U \cup \{a\}$. Connect a to every vertex of U. Connect b to every vertex of $T \cup U$. Make $\{a, b, c\}$ a triangle. Finally, for each F_i in the hitting set instance, connect t_i to every vertex $v \in F_i \subseteq U$.

($\implies$) Suppose that $D \subseteq U$ is a hitting set of size k. It can be verified that $D \cup \{a, b\}$ is a latency-2 CDS for G, i.e., that for every ordered pair of nodes (i, j) with $i \neq j$ and $(i, j) \notin E$, there is a node $v \in D \cup \{a, b\}$ such that (i, v) and (v, j) are edges in E.

($\Longleftarrow$) Now, suppose that $D \subseteq V$ is a latency-2 CDS of size $k+2$. We argue that $D \cap U$ is a hitting set of size at most k. Observe that there is no edge (c, r) and so to ensure 2-hop communication from c to r, D must contain a vertex from $N^+(c) \cap N^-(r)$, and $N^+(c) \cap N^-(r) = \{a\}$ so $a \in D$. Similarly, (c, t_1) is not an edge and $N^+(c) \cap N^-(t_1) = \{b\}$ so $b \in D$. This shows that $|D \cap U| \leq |D| - 2 = k$. Now we show that $D \cap U$ is a hitting set. Recall that, for each $i = 1, \ldots, h$, the edge (r, t_i) does not exist. So, since D is a latency-2 CDS, at least one vertex from $N^+(r) \cap N^-(t_i) = F_i \subseteq U$ must belong to D. Thus, $D \cap U$ is a hitting set of size at most k. □

Corollary 1 (Inapproximability)**.** *There is a constant $\alpha > 0$ such that it is NP-hard to approximate the minimum latency-2 CDS problem to within a factor of $\alpha \ln n$, where n refers to the number of vertices, even under bidirected edges and hop-based distances.*

Proof. This follows by Theorem 1 and the inapproximability of hitting set (Raz and Safra, 1997; Alon et al., 2006; Moshkovitz, 2012; Dinur and Steurer, 2014). □

3 Integer Programming Formulations

In what follows, we propose two IP formulations for the minimum latency-s CDS problem: CUT and POLY.

3.1 Formulation CUT

Here we propose the formulation called CUT. It has n binary variables and an exponential number of constraints—one for each (minimal) length-s vertex cut.

Definition 4 (length-s vertex cut)**.** *A subset $C \subseteq V$ of vertices is a length-s vertex cut of a directed, edge-weighted graph $G = (V, E)$ if* $\operatorname{diam}(G - C) > s$.

The correctness of formulation CUT is a consequence of the following characterization.

Proposition 3 (Characterization of latency-s CDS)**.** *A subset $D \subseteq V$ of vertices is a latency-s CDS for G if and only if $|D \cap C| \geq 1$ for every length-s vertex cut $C \subset V$.*

Proof. ($\Longrightarrow$) Assume that $D \subseteq V$ is a latency-s CDS and suppose, for sake of contradiction, that $C \subset V$ is a length-s vertex cut with $|D \cap C| = 0$. By definition of length-s vertex cut, $\operatorname{diam}(G - C) > s$, i.e., there exist vertices $a, b \in V \setminus C$ such that $\operatorname{dist}_{G-C}(a, b) > s$. By assumption that D is a latency-s CDS, there is an a-b path of length at most s whose interior vertices belong solely to D, i.e., $\operatorname{dist}_{G[D \cup \{a,b\}]}(a, b) \leq s$. Since $D \cup \{a, b\} \subseteq V \setminus C$ we have $\operatorname{dist}_{G[V \setminus C]}(a, b) \leq \operatorname{dist}_{G[D \cup \{a,b\}]}(a, b)$ which results in the following contradiction:

$$s < \operatorname{dist}_{G-C}(a, b) \triangleq \operatorname{dist}_{G[V \setminus C]}(a, b) \leq \operatorname{dist}_{G[D \cup \{a,b\}]}(a, b) \leq s.$$

($\Longleftarrow$) By the contrapositive. Suppose that $D \subseteq V$ is not a latency-s CDS, i.e., there exist vertices $a, b \in V$ such that there is no a-b path of length at most s whose interior vertices belong to D, i.e., $\operatorname{dist}_{G[D \cup \{a,b\}]}(a, b) > s$. This implies that $\operatorname{diam}(G[D \cup \{a, b\}]) > s$, and so $C := V \setminus (D \cup \{a, b\})$ is a length-s vertex cut. Moreover, $|D \cap C| = 0$, as desired. □

Proposition 3 immediately implies the correctness of the formulation CUT:

$$\min \sum_{i \in V} x_i \tag{8}$$

$$\sum_{i \in C} x_i \geq 1, \ \forall \text{ length-}s \text{ vertex cut } C \subset V \tag{9}$$

$$x_i \in \{0,1\}, \ \forall i \in V. \tag{10}$$

In general, there can be exponentially many of the constraints (9), even if we restrict ourselves to *inclusion-minimal* length-s vertex cuts. This formulation generalizes the CDS formulation based on vertex cuts that is essentially due to Fujie (2004). We address the separation complexity for constraints (9) in Section 4.

Not every valid inequality of the form $\sum_{i \in C} x_i \geq 1$ is a length-s vertex cut inequality. For example, $\sum_{i \in V \setminus \{v\}} x_i \geq 1$ is valid when $G = (V, E)$ is the bidirected 4-cycle, but $V \setminus \{v\}$ is not a length-s vertex cut. However, the following shows that the length-s vertex cut inequalities are the only *meaningful* valid inequalities of this type.

Lemma 1. *Let $C \subset V$. The inequality $|S \cap C| \geq 1$ holds for every latency-s CDS $S \subseteq V$ if and only if C is a superset of some length-s vertex cut C'.*

Proof. The 'if' direction follows easily by Proposition 3, so suppose that $|S \cap C| \geq 1$ holds for every latency-s CDS $S \subseteq V$. Let $D = V \setminus C$. By our assumption, D cannot be a latency-s CDS, i.e., there exist vertices $a, b \in V$ such that $\text{dist}_{G[D \cup \{a,b\}]}(a, b) > s$. So, $s < \text{diam}(G[D \cup \{a, b\}]) = \text{diam}(G - C')$, where $C' = V \setminus (D \cup \{a, b\})$. Thus, C' is a length-s vertex cut for G, and $C \supseteq C'$, as desired. □

Given that the minimum latency-s CDS problem admits the formulation CUT (and by Lemma 1), there are immediate polyhedral consequences (cf. Sassano (1989)), so we provide the following proposition without proof.

Proposition 4 (Basic polyhedral analysis)**.** *The convex hull of (characteristic vectors of) latency-s CDS's is full-dimensional if and only if every length-s vertex cut has size at least two. Further, if it is full-dimensional, then*

1. *for each $v \in V$,*
   (a) *$x_v \leq 1$ induces a facet;*
   (b) *$x_v \geq 0$ induces a facet if and only if v does not belong to a length-s vertex cut of size two.*
2. *for $C \subset V$, the inequality $\sum_{i \in C} x_i \geq 1$ induces a facet if and only if*
   (a) *C is a minimal length-s vertex cut, and*
   (b) *for each $v \in V \setminus C$ there exists $c \in C$ such that $(V \setminus C) \cup \{c\} \setminus \{v\}$ is a latency-s CDS.*

3.2 Generalizing the Formulation CUT for Fault-Tolerance

Here, we consider the *robust* or *fault-tolerant* variant of a latency-s CDS. That is, we are interested in a vertex subset that remains a latency-s CDS when few vertices fail.

Definition 5. *A subset $D \subseteq V$ of vertices is an r-robust latency-s CDS for graph G if, for every $F \subseteq D$ with $|F| < r$, the vertex subset $D \setminus F$ is a latency-s CDS for G.*

A consequence of Proposition 3 is the following characterization.

Corollary 2 (Characterization of r-robust latency-s CDS)**.** *A subset $D \subseteq V$ of vertices is an r-robust latency-s CDS if and only if $|D \cap C| \geq r$ for every length-s vertex cut $C \subset V$.*

Corollary 2 immediately implies the correctness of the following formulation for the minimum r-robust latency-s CDS problem:

$$\min \sum_{i \in V} x_i \tag{11}$$

$$\sum_{i \in C} x_i \geq r, \ \forall \text{ length-}s \text{ vertex cut } C \subset V \tag{12}$$

$$x_i \in \{0, 1\}, \ \forall i \in V. \tag{13}$$

This formulation generalizes previously existing formulations for the minimum k-k-CDS problem (Ahn and Park, 2015; Buchanan et al., 2015).

Figure 1(b) gives a feasible solution to this problem when $(s, r) = (3, 2)$ (using hop-based distances and treating the undirected edges as bidirected edges). That is, the gray vertices remain a latency-3 CDS when one of them fails. This is also an optimal solution for $(s, r) = (2, 1)$. However, there is no solution for $(s, r) = (2, 2)$, as evidenced by the length-2 vertex cut $C = \{5\}$. Indeed, this implies that the inequality $x_5 \geq 2$ is valid, but of course no binary vector x can satisfy this constraint.

We remark that our formalization of a fault-tolerant low-latency virtual backbone might *not* provide for r vertex-disjoint paths of length at most s (of CDS vertices) between every pair of vertices. An example is given in Figure 3, treating the undirected edges as bidirected edges. This should not be surprising given that there is, in general, no "Menger's theorem" for length-bounded paths, cf. Lovász et al. (1978).

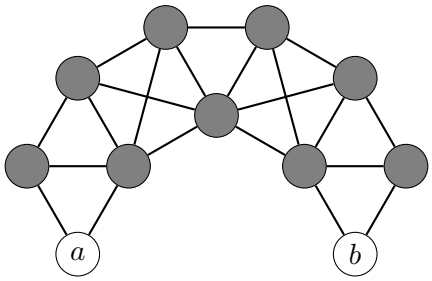


Figure 3: A CDS that maintains 5-hop communication paths when any one vertex fails, but that does not have a pair of vertex-disjoint length-5 a-b paths. Observe that this is a unit disk graph.

3.3 Formulation POLY

Here we propose the formulation called POLY. It is introduced primarily for comparison purposes and is inspired by a formulation for s-clubs given by Veremyev and Boginski (2012). It applies to hop-based case.

As before, the binary variable x_i represents the decision to include vertex i in the latency-s CDS. The binary variable y_{ij}^t equals one if and only if there exists a directed path in G from i to j of length exactly t whose interior vertices belong to the chosen CDS. This variable is only defined when $t \geq 2$ and should not be confused with y_{ij} raised to the t-th power. To formulate our problem,

we should write constraints that impose the following condition:

$$y_{ik}^t = 1 \iff \Big(\text{there exists } j \in N^-(k) \text{ such that } y_{ij}^{t-1} = 1 \text{ and } x_j = 1\Big).$$

In words, there is a path (across CDS nodes) from i to k of length t if and only if (i) there is a path (across CDS nodes) of length $t-1$ from i to some in-neighbor j of node k, and (ii) node j belongs to the CDS. When $t \geq 3$, this equivalence can be formulated as follows.

$$(\Longleftarrow) \qquad y_{ij}^{t-1} + x_j \leq y_{ik}^t + 1 \qquad \forall j \in N^-(k)$$
$$(\Longrightarrow) \qquad y_{ik}^t \leq \sum_{j \in N^-(k)} y_{ij}^{t-1} x_j.$$

The second implication is enforced via a constraint that has products of binary variables. For linearization purposes, introduce (binary) variables z_{ij}^{t-1} to replace the terms $y_{ij}^{t-1}x_j$. To impose that $z_{ij}^{t-1} = y_{ij}^{t-1}x_j$, use the usual linear constraints:

$$\begin{aligned} z_{ij}^{t-1} &\leq y_{ij}^{t-1} \\ z_{ij}^{t-1} &\leq x_j \\ y_{ij}^{t-1} + x_j &\leq z_{ij}^{t-1} + 1. \end{aligned}$$

These ideas lead to the following formulation, where the special case $t = 2$ is handled via constraints (15) and (16). Let $T_{\geq 3} := \{3, \ldots, s\}$ and $N^-[j] := N^-(j) \cup \{j\}$.

$$\min \sum_{i \in V} x_i \tag{14}$$
$$x_j \leq y_{ik}^2 \qquad j \in N^+(i) \cap N^-(k),\ i \in V \setminus \{k\},\ k \in V \tag{15}$$
$$y_{ik}^2 \leq \sum_{j \in N^+(i) \cap N^-(k)} x_j \qquad i \in V \setminus \{k\},\ k \in V \tag{16}$$
$$y_{ij}^{t-1} + x_j \leq y_{ik}^t + 1 \qquad i \in V \setminus \{j,k\},\ (j,k) \in E,\ t \in T_{\geq 3} \tag{17}$$
$$y_{ik}^t \leq \sum_{j \in N^-(k)} z_{ij}^{t-1} \qquad i \in V \setminus \{k\},\ k \in V,\ t \in T_{\geq 3} \tag{18}$$
$$z_{ij}^{t-1} \leq y_{ij}^{t-1} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{19}$$
$$z_{ij}^{t-1} \leq x_j \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{20}$$
$$y_{ij}^{t-1} + x_j \leq z_{ij}^{t-1} + 1 \qquad i \in V \setminus \{j\},\ j \in V,\ t \in T_{\geq 3} \tag{21}$$
$$\sum_{t=2}^{s} y_{ij}^t \geq 1 \qquad i \in V \setminus N^-[j],\ j \in V \tag{22}$$
$$x_i \in \{0,1\} \qquad i \in V \tag{23}$$
$$y_{ij}^t \in \{0,1\} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in \{2, \ldots, s\} \tag{24}$$
$$z_{ij}^t \in \{0,1\} \qquad i \in V \setminus \{j\},\ j \in V,\ t \in \{2, \ldots, s-1\}. \tag{25}$$

The constraints (22) ensure that there is a path of length at most s (across CDS vertices) from i to j when $(i,j) \notin E$. So, by the ideas presented above, it is straightforward to prove the following.

Theorem 2. *Under hop-based distances, the above is a correct formulation for the minimum latency-s CDS problem and has $\Theta(sn^2)$ variables, $\Theta(snm)$ constraints, and $\Theta(snm)$ nonzeros.*

Since modern MIP solvers use sparse matrix representation, this formulation's size in computer memory can be approximated by the number $\Theta(snm)$ of nonzeros. This is much less than the quantity obtained by multiplying the number of variables by the number of constraints.

Not all of these variables and constraints may be necessary. For example, if G is bidirected, we can assume that $y^t_{ij} = y^t_{ji}$. If desired, the user can impose these constraints $y^t_{ij} = y^t_{ji}$ when implementing the formulation, and the MIP solver will perform the appropriate substitutions in its presolve phase.

Based on our computational experiments, it is possible that formulation POLY is weaker than CUT, although we could not find a proof. In Section 2 of the Online Supplement, we provide a fractional point (x^*, y^*, z^*) that belongs to POLY's LP relaxation, but its x^* does not belong to CUT's LP relaxation.

4 The Complexity of the Formulations

In this section, we determine the separation complexity for the constraints defining formulation CUT and its fault-tolerant generalization. On the way, we answer an open question of Xu et al. (2005) regarding the complexity of computing a graph's fault diameter.

4.1 Computing the Fault Diameter of Graphs

As a helpful first step to determining the separation complexity, we show that a related problem, which we call Diameter Interdiction by Node Deletion, is NP-complete.

Problem: Diameter Interdiction by Node Deletion.
Input: a simple graph $G = (V, E)$ and integers q and L.
Question: Is there a subset $C \subset V$ of q vertices such that $\text{diam}(G - C) > L$?

This problem is defined for an undirected and unweighted graph G, and the diameter that is referred to is hop-based.

Theorem 3. *For each $L \geq 5$,* Diameter Interdiction by Node Deletion *is NP-complete.*

To prove this theorem, we craft reductions from Length-Bounded a-b Node Cut, which is known to be NP-complete and hard to approximate (Baier et al., 2010). Notice that this problem has specified end nodes a and b, while Diameter Interdiction by Node Deletion does not. We provide two reductions, given in Lemma 2 and Lemma 3, that together prove Theorem 3.

Problem: Length-Bounded a-b Node Cut.
Input: A simple graph $G' = (V', E')$, nonadjacent $a, b \in V'$, and integers q' and L'.
Question: Is there a subset $C' \subseteq V' \setminus \{a, b\}$ of q' vertices such that $\text{dist}_{G'-C'}(a, b) > L'$?

Lemma 2. *For each odd $L \geq 5$,* Diameter Interdiction by Node Deletion *is NP-complete.*

Proof. Membership in NP is obvious. For the reduction, consider an instance of Length-Bounded a-b Node Cut defined by graph $G' = (V', E')$, vertices $a, b \in V'$, and integers q' and odd $L' \geq 5$. Let $q = q'$ and $L = L'$. Now we construct $G = (V, E)$. The idea is to connect every pair of vertices from G' (besides a and b) by carefully adding many short paths so that the only possible way to

cheaply disrupt the diameter of G is to cut all short paths from a to b. Construct V as follows.

$$
\begin{aligned}
V &:= V' \cup T \cup A \cup B \cup W \\
T &:= \{t_i \mid 1 \le i \le q+1\} \\
A &:= \left\{ a_i^j \;\middle|\; 1 \le i \le q+1,\ 1 \le j \le \frac{L-1}{2} \right\} \\
B &:= \left\{ b_i^j \;\middle|\; 1 \le i \le q+1,\ 1 \le j \le \frac{L-1}{2} \right\} \\
W &:= \left\{ v_i^j \;\middle|\; 1 \le i \le q+1,\ 1 \le j \le \frac{L-3}{2},\ v \in V' \setminus \{a,b\} \right\}.
\end{aligned}
$$

Notice that $|V| = O(qL|V'|)$ and $q \le |V'|$ and $L \le |V'|$, so the reduction will be polynomial.

Construct the edge set E of G as follows. First, connect the vertices of V' so that $G[V'] = G'$. Then make T a clique in G. Similarly, make each $A^j := \{a_i^j \mid 1 \le i \le q+1\}$ a clique. Do the same for each $B^j := \{b_i^j \mid 1 \le i \le q+1\}$ and for each $W^j(v) := \{v_i^j \mid 1 \le i \le q+1\}$. Connect a to every vertex of A^1; and every vertex of A^1 to every vertex of A^2; and so on. Connect b to every vertex of B^1; and every vertex of B^1 to every vertex of B^2; and so on. Then for every $v \in V' \setminus \{a,b\}$, connect v to every vertex of $W^1(v)$; and every vertex of $W^1(v)$ to every vertex of $W^2(v)$; and so on. Finally, letting $p = \frac{L-1}{2}$, connect every vertex of T to every vertex of $A^p \cup B^p \cup \left(\cup_{v \in V' \setminus \{a,b\}} W^{p-1}(v)\right)$. Creating E obviously can be done in polynomial time. See Figure 4 for an illustration.

Observe that there exist at least $q+1$ (internally) node-disjoint paths of length at most L between every pair of vertices of G (the interior vertices of which belong to $V \setminus V'$), except possibly for the pair $\{a,b\}$. Moreover, (simple) a-b paths of length at most L in G can only cross vertices of V'. Thus, it can be argued that the instance (G', a, b, q', L') of Length-Bounded a-b Node Cut is a "yes" if and only if the instance (G, q, L) of Diameter Interdiction by Node Deletion is a "yes." □

Lemma 3. *For each even $L \ge 5$,* Diameter Interdiction by Node Deletion *is NP-complete.*

Proof. Membership in NP is obvious. For the reduction, consider an instance of Length-Bounded a-b Node Cut defined by graph $G' = (V', E')$, vertices $a, b \in V'$, and integers q' and even $L' \ge 5$. Let $q = q'$ and $L = L'$. Now we construct $G = (V, E)$. The main idea behind the reduction is the same as before, but the construction is slightly different. Construct V as follows.

$$
\begin{aligned}
V &:= V' \cup T \cup T' \cup W \\
T &:= \{t_i \mid 1 \le i \le q+1\} \\
T' &:= \{t_i' \mid 1 \le i \le q+1\} \\
W &:= \left\{ v_i^j \;\middle|\; 1 \le i \le q+1,\ 1 \le j \le \frac{L}{2} - 1,\ v \in V' \right\}.
\end{aligned}
$$

Notice that $|V| = O(qL|V'|)$ and $q \le |V'|$ and $L \le |V'|$, so the reduction will be polynomial.

Construct the edge set E of G as follows. First, connect the vertices of V' so that $G[V'] = G'$. Then make $T \cup T'$ a clique in G. Similarly, make each $W^j(v) := \{v_i^j \mid 1 \le i \le q+1\}$ a clique. For every $v \in V'$, connect v to every vertex of $W^1(v)$; and every vertex of $W^1(v)$ to every vertex of $W^2(v)$; and so on. Let $p = \frac{L}{2} - 1$. Connect every vertex of T to every vertex of $\cup_{v \in V' \setminus \{b\}} W^p(v)$. Similarly, connect every vertex of T' to every vertex of $\cup_{v \in V' \setminus \{a\}} W^p(v)$. Creating E obviously can be done in polynomial time. See Figure 5 for an illustration.

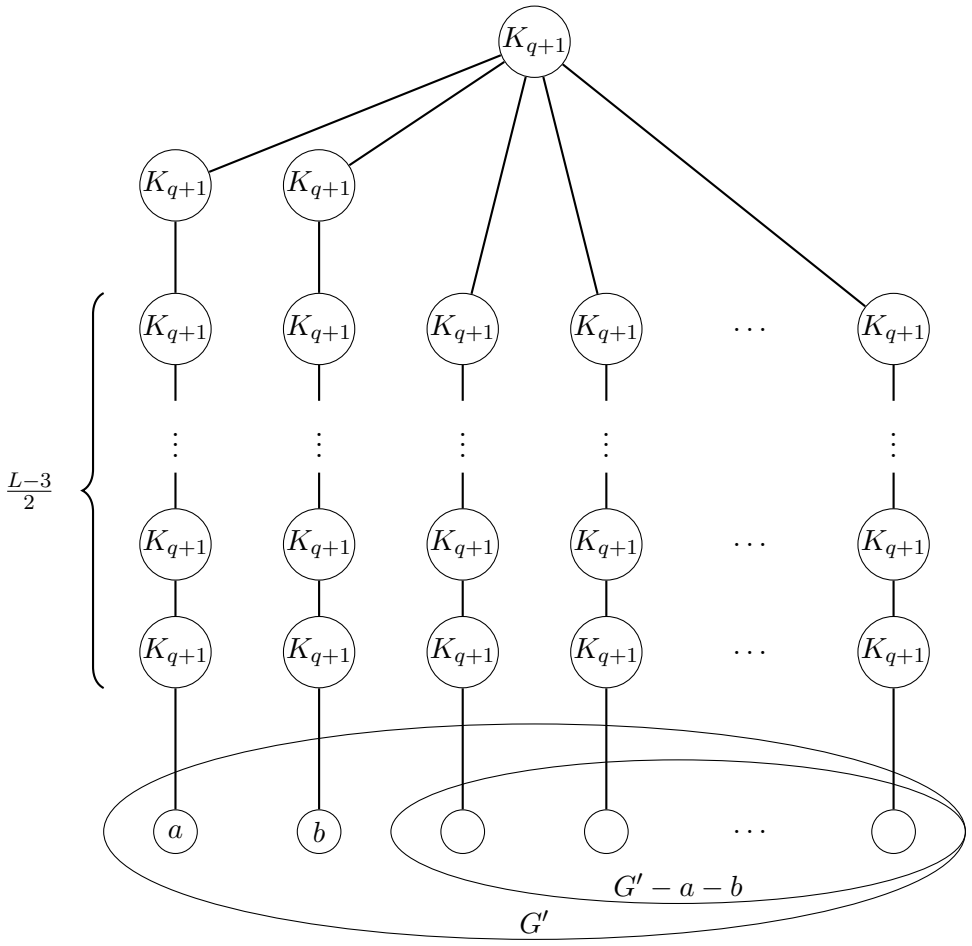


Figure 4: Illustration of the reduction for odd $L \geq 5$. Here, K_n is a complete graph on n nodes.

Observe that there exist at least $q+1$ (internally) node-disjoint paths of length at most L between every pair of vertices of G (the interior vertices of which belong to $V \setminus V'$), except possibly for the pair $\{a, b\}$. Moreover, (simple) a-b paths of length at most L in G can only cross vertices of V'. Thus, it can be argued that the instance (G', a, b, q', L') of Length-Bounded a-b Node Cut is a "yes" if and only if the instance (G, q, L) of Diameter Interdiction by Node Deletion is a "yes." □

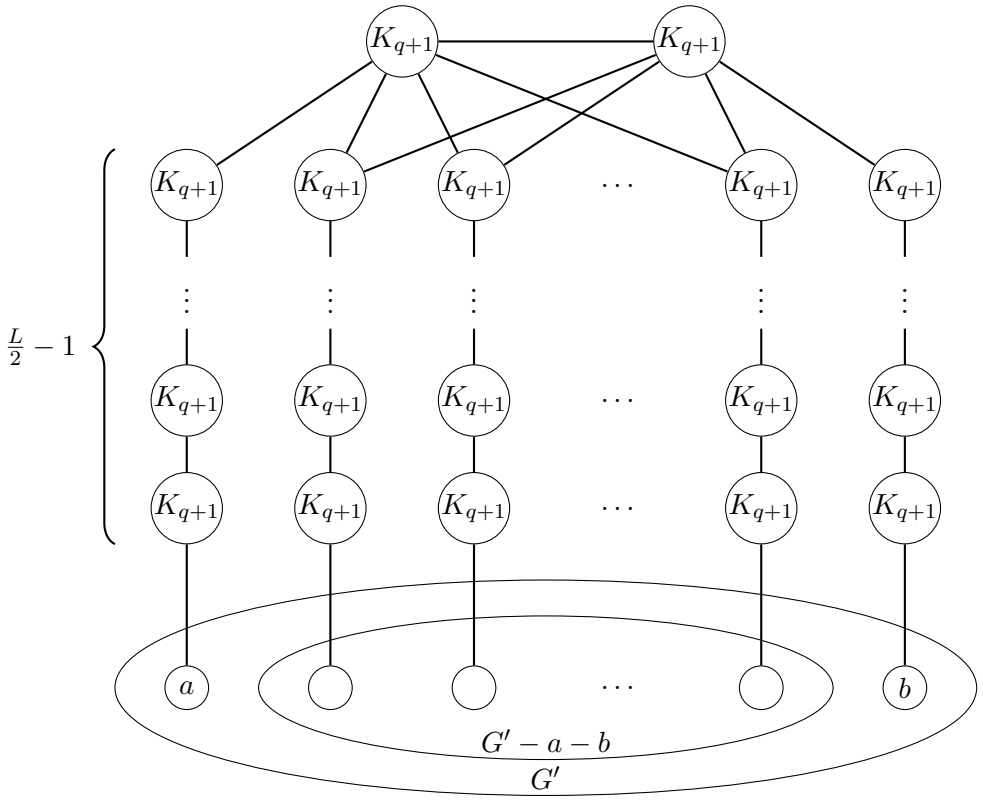


Figure 5: Illustration of the reduction for even $L \geq 5$.

Researchers have studied related notions of the *fault diameter* of a graph (Krishnamoorthy and Krishnamurthy, 1987; Xu, 2001). For example, Xu (2001) defines the f-fault diameter of graph $G = (V, E)$ to be

$$D_f(G) := \max \{\text{diam}(G - F) \mid F \subseteq V,\ |F| < f\},$$

and states that computing this value "is a quite difficult problem," but no justification is given[3]. Later, Xu et al. (2005) listed its NP-hardness as an open problem. Theorem 3 implies that computing $D_f(G)$ is indeed NP-hard when f is part of the input, say, by letting $f = q + 1$ and returning "yes" if $D_f(G) > L$.

Corollary 3. *Computing the f-fault diameter is NP-hard when f is part of the input.*

[3] Schoone et al. (1987) show a related result for increasing the diameter by *edge deletions*, but it is not clear how to modify their result for our purposes. Their definition of the problem (strangely) only allows edge deletions that maintain connectivity of the graph. This allows them to perform a reduction from Hamiltonian Path by seeking subsets of $m - (n-1)$ edges whose removal increases the diameter to $n - 1$. We feel that this is an unsatisfying hardness reduction since a minimum cut likely has fewer edges, and its removal would make the diameter infinite. In contrast, our diameter parameter can be a small constant, and we allow for arbitrary vertex deletions.

4.2 The Separation Problem for CUT

Formulation CUT has an exponential number of constraints (9), as does its fault-tolerant generalization (12), making it a nontrivial question as to how they should be used. A helpful observation, however, is that by the polynomial equivalence of optimization and separation (Grötschel et al., 1993), their LP relaxations can be solved in polynomial time if and only if their separation problems (defined below) can be solved in polynomial time.

Problem: Separation Problem for Formulation CUT.
Input: a directed and edge-weighted graph $G = (V, E)$, a weight $x_v^* \in [0, 1]$ for each $v \in V$, an integer $r \geq 1$, a number s.
Output: (if any exist) a length-s vertex cut $C \subseteq V$ with $\sum_{i \in C} x_i^* < r$.

For purposes of generality, we define this separation problem for the fault-tolerant generalization, which has right-hand-side r. We provide both positive and negative results.

Theorem 4. *Under hop-based distances, the separation problem is:*

1. *polynomial-time solvable for $s \in \{2, 3, 4\}$, for every $r \geq 1$;*
2. *(in its decision version) NP-complete for every $s \geq 5$, even when $r = 1$.*

Proof. First, we prove that item 2 holds. Membership in NP is clear, so we only show hardness. The reduction is from an instance of Diameter Interdiction by Node Deletion given by (G, q, L), which is NP-complete for each $L \geq 5$ by Theorem 3. Bidirect $G = (V, E)$ yielding directed graph $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$. Let $r = 1$, $s = L$, and $x_i^* = \frac{1}{q+1}$ for every $i \in V$. We argue that (G, q, L) is a "yes" instance of Diameter Interdiction by Node Deletion if and only if $(\overleftrightarrow{G}, x^*, r, s)$ admits a violated length-s vertex cut inequality (12). Suppose there is a violated length-s vertex cut inequality (12) for some $C \subseteq V$. Then, $\frac{|C|}{q+1} = \sum_{i \in C} x_i^* < 1$, i.e., $|C| \leq q$, and the instance of Diameter Interdiction by Node Deletion is a "yes." Now, if there is a length-s vertex cut $C' \subseteq V$ with $|C'| \leq q$ for $\overleftrightarrow{G}$, then $\sum_{i \in C'} x_i^* = \frac{|C'|}{q+1} \leq \frac{q}{q+1} < 1$ and so x^* violates the length-s vertex cut inequality $\sum_{i \in C'} x_i \geq 1$.

Now, we discuss *why* item 1 holds. In the cases $s \in \{2, 3, 4\}$, we can find a *most-violated* length-s vertex cut inequality (12) by computing, for each $(a, b) \in (V \times V) \setminus E$, a minimum-weight length-s a, b-vertex cut and comparing its weight to r. The cases $s \in \{2, 3\}$ are fairly straightforward, e.g., for $s = 2$ the solution is $N^+(a) \cap N^-(b)$. The case $s = 4$ was (essentially) shown by Lovász et al. (1978) to be polynomial-time solvable by reducing it to a particular instance of the min-cut problem, cf. Theorem 4.3.1 of Xu (2001). Since this min-cut instance can be constructed in linear time (and it is actually a subgraph of the input graph), this minimum-weight length-s a, b-vertex cut subproblem can be solved in time $O(mn)$ by Orlin (2013). Solving these subproblems for every missing edge (a, b) gives a total time of $O(mn^3)$, which is polynomial. □

By standard arguments, the flow-based separation routines referenced in the proof of Theorem 4 imply polynomial-size extended formulations for the LP relaxation of CUT when $s \in \{3, 4\}$, see Martin (1991). However, these formulations would have roughly mn^2 variables, making them too large to be practical. Hence, we do not discuss them further.

4.3 Verification and Integer Separation for the Fault-Tolerant Variant

The problem of verifying whether a given subset $D \subseteq V$ of vertices is an r-robust latency-s CDS is nontrivial. In a brute force approach, enumerate all subsets $F \subseteq D$ of $r - 1$ vertices and verify

that $D \setminus F$ is indeed a latency-s CDS. By algorithm IsLatencyConstrainedCDS, this takes time $\binom{|D|}{r-1}O(n^3) = O(n^{r+2})$, which is polynomial for any constant r. A natural question is whether this test can be performed in polynomial time when r is part of the problem input. Unfortunately, the likely answer is "no," as this is coNP-complete.

Corollary 4. *When r is part of the input, the problem of verifying whether $D \subseteq V$ is an r-robust latency-s CDS is coNP-complete for each fixed $s \geq 5$. This holds even for bidirected edges and hop-based distances.*

Proof. Membership in coNP follows because a length-s vertex cut $C \subseteq V$ with $|C| < r$ is a suitable witness when it is a "no" instance. For the reduction, consider an instance of Diameter Interdiction by Node Deletion defined by a simple graph $G = (V, E)$ and integers q and L. Bidirect its edges and let $s = L$, $r = q + 1$, and $D = V$. It can be observed that the instance of Diameter Interdiction by Node Deletion is a "yes" instance if and only if D is *not* an r-robust latency-s CDS of this bidirected graph. □

Remark 1. *As a consequence of Corollary 4, the separation problem for the constraints* (12)*, with r being part of the input, is hard* even when x^* is integer.

5 Computational Experiments

In this section, we provide results from our computational experiments. First, we demonstrate the importance of (quickly) strengthening the length-s vertex cut inequalities. Second, we provide computational results demonstrating the importance of providing an initial heuristic solution to the MIP solver. Third, we compare our full implementation of CUT with the polynomial-size formulation POLY. Our tests demonstrate the superiority of CUT over POLY. Finally, we experiment with formulation CUT for:

1. $s \in \{\text{diam}(G), \text{diam}(G) + 1, \text{diam}(G) + 2, n - 1\}$;
2. the fault-tolerant case with $r = 2$;
3. a class of instances representing node-weighted, transmitter-based delays.

All of our experiments are conducted on a Dell Precision Tower 7000 Series (7810) machine running Windows 10 enterprise, x64, with Intel® Xeon® Processor E52630 v4 (10 cores, 2.2GHz, 3.1GHz Turbo, 2133MHz, 25MB, 85W) – that is 20 logical processors – and 32 GB memory. The IP formulations were implemented in Microsoft Visual Studio 2015 in C++ for Gurobi version 7.0.2. We use default settings with the exception that we force Gurobi to use the concurrent method (which uses primal simplex, dual simplex, and barrier on different threads) for solving the root LP relaxation for POLY, as this formulation is highly degenerate and typically barrier is fastest. We impose a time limit of 3600 seconds on each instance and use the same test instances that have been used in the previous literature on the minimum CDS problem by Lucena et al. (2010); Simonetti et al. (2011); Fan and Watson (2012); Gendron et al. (2014); Buchanan et al. (2015); Li and Aneja (2017). This testbed includes both real-life and synthetically generated instances, all of which are undirected. In our experiments, we bidirect their edges.

5.1 The Importance of Strengthening the Inequalities

Since formulation CUT can have exponentially many constraints when $s \geq 3$, we initialize it with only some of the constraints. Others are added as needed via Gurobi's lazy constraint callback

features. Specifically, we start with the vertex cuts given by $N^+(i),\ i \in V$ (or inclusion-minimal subsets thereof that are also length-s vertex cuts). Then, within the branch-and-bound tree, violated length-s vertex cut inequalities are added on-the-fly.

Since the separation problem for the length-s vertex cut inequalities is NP-hard, we only separate *integer* points that the solver encounters. Each of these possible solutions $D \subseteq V$ will satisfy the initial constraints given to the solver, but D may not actually be feasible for the latency-s CDS problem. In this case, $C := V \setminus D$ is a length-s vertex cut for the graph, and the inequality $\sum_{i \in C} x_i \geq 1$ would be valid for our problem and would cut off the binary point representing D. However, this inequality is likely very weak, so we strengthen the inequality, i.e., find a minimal subset of C that is also a length-s vertex cut. This is done when initializing the formulation with the vertex cuts $N^+(i),\ i \in V$ and also when adding inequalities on-the-fly. The details are given in Section 3 of the Online Supplement. Theorem 2 of the Online Supplement shows that one can find a violated minimal length-s vertex cut inequality in time $O(n^3)$.

We also experimented with separating fractional points, particularly when $s = 3$ as this case of the separation problem is polynomial-time solvable. However, the fastest separation procedure that we are aware of takes time $O(mn^3)$ and was ultimately unhelpful—in all nine of the different implementations that we tried. See Section 4 of the Online Supplement for more details.

5.2 The Importance of Providing a Heuristic Solution to the Solver

See Section 5 of the Online Supplement.

5.3 Comparison with Formulation POLY

In Table 1, we compare the performance of CUT with that of POLY. In these tests, we set $s = \text{diam}(G)$, provide an MIP start using BestInHeuristic, and exclude instances with $s = 2$. The reason for excluding the $s = 2$ comparisons is that CUT and POLY are equally strong when $s = 2$, and so CUT will obviously perform better due to its smaller size. Since our instances are bidirected, we fix $y^t_{ij} = y^t_{ji}$ as discussed in Section 3.3.

The results demonstrate the superiority of CUT. It solves the 11 instances solved by POLY—and 9 others. Ten instances are left unsolved by both approaches; CUT provides better bounds on all of them. The formulation CUT also *quickly* solves some instances that POLY left unsolved after an hour. For example, CUT solved v50_d5, v70_d5, v70_d10, and v70_d30 each in under five seconds, while POLY solved none of them in the time limit.

5.4 The Cost of Low Latency

Imposing that a dominating set be *connected* is not too costly. Indeed, the domination number $\gamma(G)$ and the connected domination number $\gamma_c(G)$ are a constant factor apart. Specifically, they satisfy $\gamma(G) \leq \gamma_c(G) \leq 3\gamma(G) - 2$, see Haynes et al. (1998). In contrast, we show that the cost of low latency can be very large—even when decreasing the latency parameter s by one. We denote by $\gamma^{lat}_s(G)$ the size of a minimum latency-s CDS in G.

Proposition 5 (Potentially large cost of low latency)**.** *For every latency parameter $s \geq 2$, there is an infinite class of graphs G for which $\gamma^{lat}_{s+1}(G) \leq s$, but $\gamma^{lat}_s(G) \geq \Omega(n)$. This holds even when edges are bidirected and distances are hop-based.*

Proof. One such class of graphs are obtained by taking the Cartesian products $K_q \square P_s$ of a complete graph K_q and a path graph P_s and then bidirecting the edges. These graphs have sq vertices and

Table 1: A comparison of the performance of formulation CUT with that of POLY. For all graphs G, we set $s = \text{diam}(G)$ and exclude instances where $s = 2$. We report the optimal objective (or the best lower/upper bounds $[L, U]$ after one hour) under the columns labeled obj. We also give the total solve time (total), where a dash indicates > 3600 seconds.

			POLY		CUT	
graph	s	hobj	obj	total	obj	total
v30_d10	8	15	15	2.71	15	0.02
v30_d20	5	8	8	111.14	8	0.03
v30_d30	3	8	8	12.59	8	0.10
v50_d5	14	32	[31,32]	-	32	0.07
v50_d10	5	20	18	2711.41	18	0.18
v50_d20	3	14	14	6.31	14	0.11
v50_d30	3	8	8	398.41	8	1.12
v70_d5	8	36	[26,36]	-	32	0.53
v70_d10	4	31	[28,29]	-	29	2.98
v70_d20	3	18	[11,18]	-	17	305.96
v70_d30	3	8	[6,7]	-	7	3.33
v100_d5	5	57	[42,57]	-	56	17.91
v100_d10	4	31	[13,31]	-	[22,26]	-
v100_d20	3	20	[9,20]	-	[14,20]	-
v120_d5	6	40	[16,40]	-	31	1087.17
v120_d10	3	68	63	1522.53	63	10.31
v120_d20	3	21	[7,21]	-	[10,21]	-
v120_d30	3	12	[4,12]	-	[7,12]	-
v150_d5	5	54	[19,54]	-	[30,54]	-
v150_d10	3	65	[35,65]	-	[43,61]	-
v150_d20	3	22	[6,22]	-	[9,22]	-
v200_d5	4	92	[43,92]	-	[49,92]	-
v200_d10	3	64	[24,64]	-	[26,64]	-
v200_d20	3	22	[6,22]	-	[8,22]	-
IEEE-14	5	5	5	0.08	5	0.01
IEEE-30	6	14	14	0.29	14	0.01
IEEE-57	12	35	35	57.65	35	0.04
RTS-96	13	40	[35,39]	-	37	0.14
IEEE-118	14	48	48	130.70	48	0.15
IEEE-300	24	139	[6,139]	-	135	11.98

diameter s when $q \geq 2$. These graphs $K_q \Box P_s$ can be defined as having vertex set $V^1 \cup \cdots \cup V^s$, where each $V^i = \{v_1^i, \ldots, v_q^i\}$. For the edges, let each V^i be a clique, and connect each vertex v_j^i to its counterpart v_j^{i+1} from the next V^{i+1}. Bidirect all edges.

See that $\gamma_{s+1}^{lat}(K_q \Box P_s) \leq s$, since the s vertices v_1^i form a feasible solution. Now we show that $\gamma_s^{lat}(K_q \Box P_s) \geq \Omega(n)$ in two cases. When $s = 2$, the q vertex subsets $\{v_i^1, v_{i+1}^2\}$ for $i = 1, \ldots, q-1$ and $\{v_q^1, v_1^2\}$ form length-2 cuts and are disjoint. Thus, $\gamma_2^{lat}(K_q \Box P_2) \geq q = n/2$. When $s \geq 3$, each vertex v_j^i with $2 \leq i \leq s-1$ is a length-s vertex cut on its own (by the resulting distance between nodes v_j^1 and v_j^s), so $\gamma_s^{lat}(K_q \Box P_s) \geq (s-2)q \geq n/3$. □

We observe the cost of low latency "in practice" through the computational results given in Table 2. We report the solution sizes and runtimes for different values of the latency parameter $s \in \{\text{diam}, \text{diam}+1, \text{diam}+2, n-1\}$ under hop-based distances. Thus, we have the strictest case of $s = \text{diam}$ and the most relaxed value of $s = n-1$, which corresponds to the minimum CDS problem when edges are bidirected. The solve times tend to improve as s increases, and we are able to solve all instances when $s = n-1$. However, this is not universally the case, e.g., for graph v100_d5. Some of the lower bounds can immediately be improved based on the table. For example, we can claim a lower bound of 10 for the instance v150_d20 when $s = \text{diam}$, since 10 is optimal for the less restrictive case $s = \text{diam}+1$.

The runtimes for the case $s = n-1$ closely resemble those given by Buchanan et al. (2015) for the minimum CDS problem. This is unsurprising given that the approach taken here is very similar. However, the instance IEEE-300 takes longer here (514.34 vs. 52.88 seconds). This can be attributed to the 492.86 seconds spent in our slower callback routines.

In some applications, achieving low latency is desirable but should not be viewed as a "hard" constraint. In this case, the tradeoff between CDS size and the latency guarantee should be considered. For example, the results for graphs v30_d10 and IEEE-14 show that low latency comes for free; there is a minimum CDS that also satisfies the most restrictive (but feasible) latency value $s = \text{diam}$. On the other hand, for the graphs v100_d30 and v150_d30, the optimal objective triples when tightening the latency parameter from $s = 3$ to $s = 2$ and may not be justified.

We also give the optimal objectives for the dominating $(s-2)$-club problem. As observed in the introduction, the formalization based on dominating $(s-2)$-clubs does not quite capture the intent of the latency constraints, and here we see that it usually gives the impression that no suitable low-latency CDS exists, and yet there exists a latency-s CDS. This occurs for 37 of the 47 graphs when $s = \text{diam}(G)$. An example is given in Figure 6(a). Also, Figure 6(b) shows an instance where both problems are feasible but have different optimal solutions.

5.5 Results for the Fault-Tolerant Variant

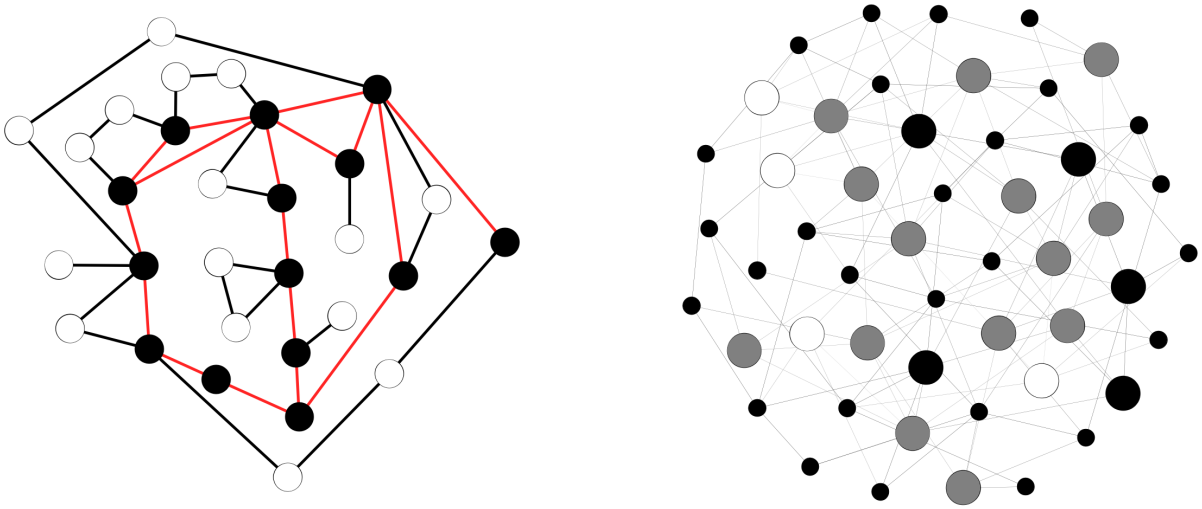
See Section 6 of the Online Supplement.

5.6 Results when Delays are Node-Weighted and Transmitter-Based

In this section, we provide computational results for instances in which delays are node-weighted and transmitter-based. The intent is to model wireless sensor networks in which delays *depend on the transmitting node*. In our experiments, we make the simplifying assumption that the delay at node i is a given constant w_i. This is the time for node i to pass a message to any neighboring node. Following the transformation given in the introduction, this means that the edges pointing away from node i should have weight w_i. However, if one were to look at our implementation, they

Table 2: Results for different values of the latency parameter s. The case where $s = n-1$ is identical to the minimum strongly connected dominating set problem. We also report the objective of the dominating $(s-2)$-club problem (club). Here, ∞ denotes infeasibility, and blank cells indicate that $s - 2 \leq 0$.

		$s = \text{diam}$			$s = \text{diam}+1$			$s = \text{diam}+2$			$s = n-1$	
graph	diam	club	obj	total	club	obj	total	club	obj	total	obj	total
v30_d10	8	∞	15	0.02	15	15	0.02	15	15	0.02	15	0.04
v30_d20	5	8	8	0.03	7	7	0.01	7	7	0.02	7	0.01
v30_d30	3	∞	8	0.10	5	5	0.05	4	4	0.01	4	0.01
v30_d50	2		7	0.01	3	3	0.01	3	3	0.01	3	0.01
v30_d70	2		3	0.04	2	2	0.01	2	2	0.01	2	0.01
v50_d5	14	32	32	0.07	32	32	0.13	31	31	0.19	31	0.36
v50_d10	5	19	18	0.18	14	14	0.18	13	13	0.11	12	0.23
v50_d20	3	∞	14	0.11	7	7	0.27	7	7	0.17	7	0.17
v50_d30	3	∞	8	1.12	5	5	0.10	5	5	0.12	5	0.12
v50_d50	2		9	0.17	3	3	0.10	3	3	0.02	3	0.02
v50_d70	2		4	0.79	2	2	0.03	2	2	0.02	2	0.02
v70_d5	8	32	32	0.53	29	29	0.74	28	28	1.38	27	0.76
v70_d10	4	∞	29	2.98	17	16	8.87	13	13	0.47	13	0.10
v70_d20	3	∞	17	305.96	8	8	0.21	7	7	0.19	7	0.14
v70_d30	3	∞	7	3.33	5	5	0.17	5	5	0.16	5	0.16
v70_d50	2		10	0.56	3	3	0.05	3	3	0.05	3	0.05
v70_d70	2		5	1.57	2	2	0.10	2	2	0.06	2	0.06
v100_d5	5	∞	56	17.91	40	[34,36]	-	29	29	2018.43	24	0.56
v100_d10	4	∞	[22,26]	-	15	15	4.24	14	14	0.41	13	0.25
v100_d20	3	∞	[14,20]	-	9	9	2.35	8	8	0.53	8	0.51
v100_d30	2		39	5.00	∞	[7,12]	-	6	6	0.94	6	0.98
v100_d50	2		12	61.27	4	4	0.87	4	4	0.89	4	0.93
v100_d70	2		5	8.17	3	3	1.20	3	3	1.18	3	1.20
v120_d5	6	31	31	1087.17	28	28	97.26	26	26	4.36	25	0.62
v120_d10	3	∞	63	10.31	∞	[17,31]	-	15	15	202.51	13	0.69
v120_d20	3	∞	[10,21]	-	9	9	5.67	8	8	3.46	8	1.54
v120_d30	3	∞	[7,12]	-	6	6	1.10	6	6	1.20	6	1.10
v120_d50	2		[11,12]	-	4	4	4.78	4	4	3.71	4	3.63
v120_d70	2		5	29.67	3	3	2.06	3	3	2.16	3	2.08
v150_d5	5	∞	[30,54]	-	[28,33]	[26,40]	-	[26,28]	[26,33]	-	26	2.10
v150_d10	3	∞	[43,61]	-	∞	[14,28]	-	16	[15,18]	-	14	4.94
v150_d20	3	∞	[9,22]	-	10	10	379.34	9	9	7.13	9	6.88
v150_d30	2		[35,41]	-	∞	[6,11]	-	6	6	7.65	6	3.96
v150_d50	2		[9,13]	-	4	4	2.38	4	4	2.49	4	2.77
v150_d70	2		6	569.09	3	3	3.29	3	3	3.35	3	3.41
v200_d5	4	∞	[49,92]	-	[31,52]	[26,50]	-	[25,46]	[26,35]	-	27	10.00
v200_d10	3	∞	[26,64]	-	∞	[14,29]	-	[14,19]	[14,21]	-	16	301.83
v200_d20	3	∞	[8,22]	-	10	[9,11]	-	9	9	183.40	9	184.27
v200_d30	2		[27,44]	-	∞	[6,12]	-	7	7	223.04	7	205.41
v200_d50	2		[8,15]	-	4	4	145.90	4	4	7.44	4	7.84
v200_d70	2		[4,7]	-	3	3	6.43	3	3	6.29	3	6.59
IEEE-14	5	5	5	0.01	5	5	0.01	5	5	0.01	5	0.01
IEEE-30	6	∞	14	0.01	13	13	0.01	11	11	0.01	11	0.01
IEEE-57	12	35	35	0.04	31	31	0.08	31	31	0.19	31	1.88
RTS-96	13	37	37	0.14	35	35	0.46	34	34	0.38	32	1.90
IEEE-118	14	48	48	0.15	46	46	0.25	45	45	0.25	43	1.18
IEEE-300	24	135	135	11.98	131	131	35.08	130	130	66.30	129	514.34



(a) IEEE-30 **(b) v50_d10**

Figure 6: Side (a) shows a minimum latency-6 CDS for IEEE-30; there is no dominating 4-club. Side (b) shows a minimum latency-5 CDS D and a minimum dominating 3-club D' for the graph v50_d10. Nodes in $D \cup D'$ are larger; nodes in $D \setminus D'$ are white; nodes in $D' \setminus D$ are black; nodes in $D \cap D'$ are gray.

would see that our weights are stored by node. We prefer this representation since it is more space efficient.

To ensure that the node-based delays w_i used in our experiments are somewhat reasonable and reproducible, we define them as follows, where $\text{dist}(i, j)$ is hop-based.

$$w_i := \left\lfloor \frac{1000(n-1)}{\sum_{j \in V} \text{dist}(i,j)} \right\rfloor .$$

The reasoning for defining w_i in this way is as follows. So-called *central* nodes in the network will be used more frequently to transmit information, resulting in longer queueing delays. The quantity $(n-1)/\sum_{j \in V} \text{dist}(i,j)$ is the definition of (normalized) closeness centrality and the first definition for w_i that we tried. However, it is fractional, and the inexactness of later floating point calculations caused problems. To avoid exact rational arithmetic, we wanted to round closeness centrality to an integer, but this would give either a zero or a one. Multiplying by a large integer (1000 in our case) allowed for more diverse delays.

Table 3 provides our results with these delays where $s = \text{diam}(G)$ is set to be as restrictive as possible while maintaining feasibility. They indicate a strength of our approach—that using weighted distances has little impact on the performance.

Table 3: Results for the weighted-distance variant. See Section 5.6 for weighting information.

graph	s	BB node	hobj	obj	total
v30_d10	2281	0	21	21	0.02
v30_d20	2317	31	12	11	0.04
v30_d30	1751	117	19	18	0.07
v30_d50	1422	48	10	9	0.06
v30_d70	1585	0	8	7	0.04
v50_d5	2549	21	37	37	0.11
v50_d10	1961	377	29	29	0.29
v50_d20	1614	158	33	32	0.21
v50_d30	1743	6317	25	22	1.17
v50_d50	1400	523	14	14	0.27
v50_d70	1606	285	8	6	0.21
v70_d5	2055	919	50	48	0.71
v70_d10	1701	42	52	50	0.51
v70_d20	1624	10527	43	40	2.37
v70_d30	1716	3556	35	30	1.47
v70_d50	1404	5631	17	15	3.04
v70_d70	1581	328	8	8	0.47
v100_d5	1701	10376	79	76	3.37
v100_d10	1785	1820846	55	[47,51]	-
v100_d20	1678	290399	34	[21,33]	-
v100_d30	1211	13773	51	47	6.46
v100_d50	1378	244673	19	17	61.96
v100_d70	1571	1250	9	8	1.82
v120_d5	1948	542530	59	53	774.43
v120_d10	1471	97436	76	70	31.07
v120_d20	1659	2327430	47	[37,43]	-
v120_d30	1695	4844453	47	40	1101.31
v120_d50	1389	6911583	18	[14,16]	-
v120_d70	1559	475	10	9	3.21
v150_d5	1757	4407299	89	87	2378.46
v150_d10	1469	1686790	102	[81,92]	-
v150_d20	1663	337800	50	[30,47]	-
v150_d30	1209	4540949	58	[43,50]	-
v150_d50	1371	4427905	24	[18,21]	-
v150_d70	1559	317	12	10	4.38
v200_d5	1594	374407	137	[80,135]	-
v200_d10	1503	419384	122	[83,109]	-
v200_d20	1640	2184837	106	[84,96]	-
v200_d30	1201	1162600	67	[45,63]	-
v200_d50	1361	4196920	26	[20,23]	-
v200_d70	1557	8868	12	11	46.41
IEEE-14	2154	0	8	8	0.01
IEEE-30	2121	0	16	16	0.02
IEEE-57	2306	0	41	41	0.11
RTS-96	2241	285	42	41	0.51
IEEE-118	2556	0	48	48	0.84
IEEE-300	2646	6558	141	137	66.90

6 Conclusion

In this paper, we introduce a latency-constrained variant of the minimum CDS problem motivated by applications in wireless sensor networks in which one seeks a virtual backbone that provides for small end-to-end delays. We propose integer programming formulations based on length-bounded vertex cuts. These formulations generalize the best-performing existing formulations for the minimum CDS problem and also generalize the best-performing formulations for the fault-tolerant variant—the minimum k-connected k-dominating set problem. A branch-and-cut implementation of formulation CUT makes easy work of synthetic instances having fewer than 100 nodes and real-life instances with up to 300 nodes, significantly outperforming formulation POLY.

Our proposed formulations are in the same vein as the recent "thin" approaches for other optimization problems (Fischetti et al., 2017, 2016). In ongoing and future research, we study the potential of using similar thin formulations based on length-bounded vertex cuts for other distance-constrained problems in networks, e.g., Buchanan and Salemi (2018).

In this paper, we focused on exact approaches. Only because our MIP solver Gurobi had problems finding feasible solutions in an hour did we employ a simple construction heuristic. There is certainly room for improvement on this front, although the negative results given in Section 1 of the Online Supplement should not be ignored.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1662757. We thank three anonymous reviewers whose comments helped to significantly improve the presentation.

References

Namsu Ahn and Sungsoo Park. An optimization algorithm for the minimum k-connected m-dominating set problem in wireless sensor networks. *Wireless Networks*, 21(3):783–792, 2015.

Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.

Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondřej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms (TALG)*, 7(1):4, 2010.

Austin Buchanan and Hosseinali Salemi. Parsimonious formulations for low-diameter clusters, 2018. `http://www.optimization-online.org/DB_HTML/2017/09/6196.html`.

Austin Buchanan, Je Sang Sung, Vladimir Boginski, and Sergiy Butenko. On connected dominating sets of restricted diameter. *European Journal of Operational Research*, 236(2):410–418, 2014.

Austin Buchanan, Je Sang Sung, Sergiy Butenko, and Eduardo L Pasiliao. An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing*, 27(1): 178–188, 2015.

Si Chen, Ivana Ljubić, and Srinivasacharya Raghavan. The regenerator location problem. *Networks*, 55(3):205–220, 2010.

Si Chen, Ivana Ljubić, and S Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220, 2015.

Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

Ding-Zhu Du and Peng-Jun Wan. *Connected Dominating Set: Theory and Applications*, volume 77 of *Springer Optimization and Its Applications*. Springer, New York, 2013.

Jack Edmonds and Ellis L Johnson. Matching: A well-solved class of integer linear programs. In R. Guy, H. Hanani, N. Sauer, and Schönheim, editors, *Proceedings Calgary International Conference on Combinatorial Structures and Their Applications*, Combinatorial structures and their applications, pages 89–92. Gordon and Breach, 1970.

Neng Fan and Jean-Paul Watson. Solving the connected dominating set problem and power dominating set problem by integer programming. In G. Lin, editor, *Combinatorial Optimization and Applications*, volume 7402 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2012.

Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4): 634–652, 1998.

Matteo Fischetti, Ivana Ljubić, and Markus Sinnl. Redesigning Benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2016.

Matteo Fischetti, Markus Leitner, Ivana Ljubić, Martin Luipersbeck, Michele Monaci, Max Resch, Domenico Salvagnin, and Markus Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017.

Tetsuya Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43 (4):212–223, 2004.

Michael R Garey and David S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

Bernard Gendron, Abilio Lucena, Alexandre Salles da Cunha, and Luidi Simonetti. Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26(4):645–657, 2014.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993.

Teresa W Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

Mukkai S Krishnamoorthy and Balaji Krishnamurthy. Fault diameter of interconnection networks. *Computers & Mathematics with Applications*, 13(5):577–582, 1987.

Deying Li, Hongwei Du, Peng-Jun Wan, Xiaofeng Gao, Zhao Zhang, and Weili Wu. Construction of strongly connected dominating sets in asymmetric multihop wireless networks. *Theoretical Computer Science*, 410(8-10):661–669, 2009.

Xiangyong Li and Yash P Aneja. Regenerator location problem: Polyhedral study and effective branch-and-cut algorithms. *European Journal of Operational Research*, 257(1):25–40, 2017.

Yingshu Li, Donghyun Kim, Feng Zou, and Ding-Zhu Du. Constructing connected dominating sets with bounded diameters in wireless networks. In *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, pages 89–94. IEEE, 2007.

László Lovász, Víctor Neumann-Lara, and Michael Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–276, 1978.

Abilio Lucena, Nelson Maculan, and Luidi Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

Richard Kipp Martin. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3):119–128, 1991.

Mike J Morgan and Vic Grout. Finding optimal solutions to backbone minimisation problems using mixed integer programming. In *Proceedings of the 7th International Network Conference (INC 2008)*, pages 53–64, 2008.

Dana Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 276–287. Springer, 2012.

James B Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774. ACM, 2013.

Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

Antonio Sassano. On the facial structure of the set covering polytope. *Mathematical Programming*, 44(1-3):181–202, 1989.

Oliver Schaudt. On dominating sets whose induced subgraphs have a bounded diameter. *Discrete Applied Mathematics*, 161(16):2647–2652, 2013.

Anneke A Schoone, Hans L Bodlaender, and Jan Van Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987.

Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, Berlin, 2003.

Luidi Simonetti, Alexandre Salles Da Cunha, and Abilio Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In *Network Optimization*, pages 162–169. Springer Berlin Heidelberg, 2011.

Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 3(69):330–353, 2004.

Alexander Veremyev and Vladimir Boginski. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research*, 218(2):316–326, 2012.

Min Xie and Martin Haenggi. Towards an end-to-end delay analysis of wireless multihop networks. *Ad Hoc Networks*, 7(5):849–861, 2009.

Junming Xu. *Topological structure and analysis of interconnection networks*. Kluwer, 2001.

Min Xu, Jun-Ming Xu, and Xin-Min Hou. Fault diameter of Cartesian product graphs. *Information Processing Letters*, 93(5):245–248, 2005.

Ning Zhang, Incheol Shin, Feng Zou, Weili Wu, and My Tra Thai. Trade-off scheme for fault tolerant connected dominating sets on size and diameter. In *Proceedings of the 1st ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*, pages 1–8. ACM, 2008.

Yi Zhong, Martin Haenggi, Fu-Chun Zheng, Wenyi Zhang, Tony QS Quek, and Weili Nie. Towards a tractable delay analysis in ultradense networks. *IEEE Communications Magazine*, page 104, 2017.

Online Supplement for "The optimal design of low-latency virtual backbones"

Hamidreza Validi and Austin Buchanan

School of Industrial Engineering & Management
Oklahoma State University
{hamidreza.validi,buchanan}@okstate.edu

June 3, 2019

1 SETH-Based Hardness Results

Below, we consider fundamental algorithmic questions about latency-s CDS's. For example, how quickly can one verify that a vertex subset is indeed a latency-s CDS? How quickly can one perform local search moves for the minimum latency-s CDS problem? These problems admit relatively straightforward algorithms that run in time $O(mn)$ when distances are hop-based. A natural question is whether there exist faster algorithms. We show that, based on the strong exponential time hypothesis (SETH), this would be difficult.

SETH is an unproven complexity assumption that is stronger than P≠NP. While it is unproven and some doubt that it is true, it is nevertheless a benchmark for gauging how surprising or noteworthy a faster algorithm would be. The reader is referred to the survey of Lokshtanov et al. (2013) for more information about SETH. To prove our results, we use the following theorem of Roditty and Vassilevska Williams (2013).

Theorem 1. *If SETH holds, then for every $\varepsilon > 0$ there is no algorithm for verifying that a simple, connected graph $G = (V, E)$ has* $\operatorname{diam}(G) = 2$ *that runs in time $O(m^{2-\varepsilon})$.*

Below, we restate and prove Proposition 2 from the main body of the paper.

Proposition 1 (restatement of Proposition 2)**.** *If SETH holds, then for every $\varepsilon > 0$ there exists no algorithm for verifying that a subset D of vertices is a latency-s CDS that runs in time $O(m^{2-\varepsilon})$, even in the simplest nontrivial case of hop-based distances and $s = 2$.*

Proof. The proof follows by reduction from the problem in Theorem 1. Namely, bidirect the edges of the graph to get $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ and let $s = 2$, $D = V$, and $w_e = 1$ for each $e \in \overleftrightarrow{E}$. It can be observed that D is a latency-s CDS for $\overleftrightarrow{G}$ if and only if G has $\operatorname{diam}(G) = 2$, and this reduction runs in linear time, so the proposition follows. □

A similar negative result shows a difficulty that would be encountered when applying local search to the minimum latency-s CDS problem. In the following proposition, we refer to the local search move in which one is given a known-to-be-feasible solution $D \subseteq V$ along with a vertex $v \in D$, and the task is to determine whether one can move to $D \setminus \{v\}$ and maintain feasibility.

Proposition 2. *If SETH holds, then for every $\varepsilon > 0$ there exists no algorithm for the local search move defined above that runs in time $O(m^{2-\varepsilon})$, even in the simplest nontrivial case of hop-based distances and $s = 2$.*

Proof. The proof follows by reduction from the problem in Theorem 1, where we assume, without loss, that G is not complete and thus $\text{diam}(G) \geq 2$. Add a new node v to the graph $G = (V, E)$ and connect it to all other nodes. Call this new graph $G' = (V', E')$, where $V' = V \cup \{v\}$ and $E' = E \cup \{\{u, v\} \mid u \in V\}$. Bidirect all edges of G' to get $\overleftrightarrow{G}' = (V', \overleftrightarrow{E}')$. Let $s = 2$ and $D = V'$ and $w_e = 1$ for each $e \in \overleftrightarrow{E}'$. Since v is an in-neighbor and an out-neighbor of all other nodes in $\overleftrightarrow{G}'$ and since $v \in D$, it is clear that D is a latency-s CDS for $\overleftrightarrow{G}'$. It can be observed that $D \setminus \{v\}$ is a latency-s CDS for $\overleftrightarrow{G}'$ if and only if G has $\text{diam}(G) = 2$, and this reduction runs in linear time, so the proposition follows. □

2 Strength of CUT vs. POLY

Figure 1 shows that POLY cannot be stronger than CUT. We were unable to prove/disprove that CUT is stronger than POLY.

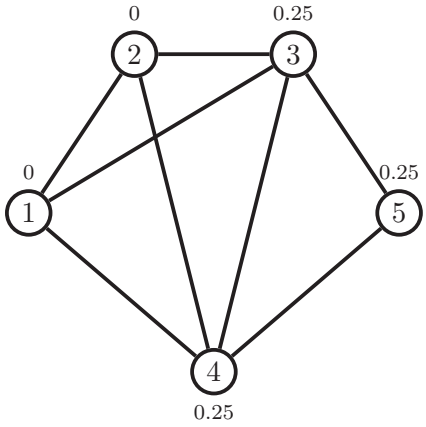


Figure 1: A graph with nodes numbered $\{1, 2, 3, 4, 5\}$ and values for x^* given by the nodes. When the edges shown are treated as bidirected and unit-weighted, this vector x^* is infeasible for formulation CUT (for every s), since the length-s vertex cut inequality for $C = \{3, 4\}$ is violated. However, there exist values y^* and z^* for which (x^*, y^*, z^*) satisfies formulation POLY when $s = 3$. Indeed, these values for x^* are *optimal* for POLY's LP relaxation. The solve logs, obtained via Gurobi, are below.

```
Optimize a model with 230 rows, 80 columns and 590 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 131 rows and 45 columns
Presolve time: 0.00s
Presolved: 99 rows, 35 columns, 241 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.000000e+00   0.000000e+00      0s
      20    5.0000000e-01   0.000000e+00   0.000000e+00      0s
Solved in 20 iterations and 0.00 seconds
Optimal objective  5.000000000e-01

X_1 = 0
X_2 = 0
X_3 = 0.25
X_4 = 0.25
X_5 = 0
Y_{1,2}^2 = 0.25
Y_{1,3}^2 = 0.25
Y_{1,4}^2 = 0.25
```

```
Y_{1,5}^2 = 0.5
Y_{2,1}^2 = 0.25
Y_{2,3}^2 = 0.25
Y_{2,4}^2 = 0.25
Y_{2,5}^2 = 0.5
Y_{3,1}^2 = 0.25
Y_{3,2}^2 = 0.25
Y_{3,4}^2 = 0
Y_{3,5}^2 = 0.25
Y_{4,1}^2 = 0.25
Y_{4,2}^2 = 0.25
Y_{4,3}^2 = 0
Y_{4,5}^2 = 0.25
Y_{5,1}^2 = 0.5
Y_{5,2}^2 = 0.5
Y_{5,3}^2 = 0.25
Y_{5,4}^2 = 0.25
Y_{1,2}^3 = 0
Y_{1,3}^3 = 0
Y_{1,4}^3 = 0
Y_{1,5}^3 = 0.5
Y_{2,1}^3 = 0
Y_{2,3}^3 = 0
Y_{2,4}^3 = 0
Y_{2,5}^3 = 0.5
Y_{3,1}^3 = 0
Y_{3,2}^3 = 0
Y_{3,4}^3 = 0
Y_{3,5}^3 = 0
Y_{4,1}^3 = 0
Y_{4,2}^3 = 0
Y_{4,3}^3 = 0
Y_{4,5}^3 = 0
Y_{5,1}^3 = 0.5
Y_{5,2}^3 = 0.5
Y_{5,3}^3 = 0
Y_{5,4}^3 = 0
Z_{1,2}^2 = 0
Z_{1,3}^2 = 0
Z_{1,4}^2 = 0
Z_{1,5}^2 = 0
Z_{2,1}^2 = 0
Z_{2,3}^2 = 0
Z_{2,4}^2 = 0
Z_{2,5}^2 = 0
Z_{3,1}^2 = 0.25
Z_{3,2}^2 = 0.25
Z_{3,4}^2 = 0
```

```
Z_{3,5}^2 = 0.25
Z_{4,1}^2 = 0.25
Z_{4,2}^2 = 0.25
Z_{4,3}^2 = 0
Z_{4,5}^2 = 0.25
Z_{5,1}^2 = 0
Z_{5,2}^2 = 0
Z_{5,3}^2 = 0
Z_{5,4}^2 = 0
```

3 The Importance of Strengthening the Inequalities

The following algorithm can be used to strengthen length-s vertex cut inequalities, i.e., to find *inclusion-minimal* length-s vertex cuts. For our purposes, it is best described in terms of a "bad" vertex subset $B \subseteq V$, i.e., one that is not a latency-s CDS for G. (If given a length-s vertex cut, initialize the algorithm using its complement.) The algorithm takes B as input and returns an inclusion-minimal length-s vertex cut $C \subseteq V \setminus B$ for G.

MinimalizeBasic(G, B, s):

1. $Q \leftarrow V \setminus B$;
2. for $q \in Q$ do
   - if $B \cup \{q\}$ is **not** a latency-s CDS for G then update $B \leftarrow B \cup \{q\}$;
3. return $C := V \setminus B$.

Proposition 3. *Algorithm* MinimalizeBasic *finds a minimal length-s vertex cut in time $O(n^4)$.*

Proof. The runtime is dominated by the $|Q|$ different calls to IsLatencyConstrainedCDS, which take time $O(|Q|n^3) = O(n^4)$. Since latency-s CDS's are closed under taking supersets (as a consequence of Proposition 3 of the paper), it can be argued that, in step 3, $B \cup \{v\}$ is a latency-s CDS for every $v \in V \setminus B$. Further, B is not. Thus, $|D \cap C| \geq 1$ for every latency-s CDS $D \subseteq V$, and no proper subset of C satisfies this property. So, by Lemma 1 of the paper, C is a minimal length-s vertex cut. □

The algorithm MinimalizeBasic for finding a minimal length-s vertex cut runs in time $O(n^4)$. Here, we provide an improved $O(n^3)$ implementation. It is based on maintaining the collection F of "far" pairs of vertices, i.e., pairs of vertices that cannot communicate quickly enough through $B \subseteq V$.

First we provide pseudocode for the subroutine EnumerateFarPairs, which finds all far pairs. It is similar to IsLatencyConstrainedCDS described earlier but requires more space, since it requests the set F (and not just a query as to whether F is empty). Recall the subgraph of G denoted $\overrightarrow{G}_v^B$ whose edge set $\overrightarrow{E}_v^B$ was defined in equation 7 of the paper. This is used to compute shortest paths from v to all other nodes, using nodes from B as intermediaries.

EnumerateFarPairs(G, B, s):

1. $F \leftarrow \emptyset$;
2. for each $v \in V$ do
   - compute shortest paths from v in digraph $\overrightarrow{G}_v^B$;
   - for $t \in V \setminus \{v\}$ do
     - if $\text{dist}_{\overrightarrow{G}_v^B}(v,t) > s$ then $F \leftarrow F \cup \{(v,t)\}$;
3. return F.

Now we give an improved implementation of MinimalizeBasic. In its pseudocode, we need notation for another subgraph of G, denoted $\overleftarrow{G}_v^B = (V, \overleftarrow{E}_v^B)$, that is used when computing the shortest paths from all other nodes to v, using only nodes from B as intermediaries. Its edge set $\overleftarrow{E}_v^B$ contains all edges that might be used in one of these paths.

$$\overleftarrow{E}_v^B := E(B) \cup \delta^-(B) \cup \left(\delta^+(B) \cap \delta^-(v)\right). \tag{1}$$

Minimalize(G, B, s):

1. $Q \leftarrow V \setminus B$ and $F \leftarrow$EnumerateFarPairs(G, B, s);
2. for $v \in Q$ do
   - compute shortest paths from v in graph $\overrightarrow{G}_v^B$;
   - compute shortest paths to v in graph $\overleftarrow{G}_v^B$;
   - $R \leftarrow \{(a,b) \in F \mid \text{dist}_{\overleftarrow{G}_v^B}(a,v) + \text{dist}_{\overrightarrow{G}_v^B}(v,b) \leq s\}$;
   - if $F \neq R$ then update $B \leftarrow B \cup \{v\}$ and $F \leftarrow F \setminus R$;
3. return $C := V \setminus B$.

Here, $R \subseteq F$ represents the subset of far pairs that would no longer be far if paths could cross node v. If the sets R and F are the same, then this means that $B \cup \{v\}$ a latency-s CDS (which we do not want), otherwise the algorithm adds v to B, thus moving to the larger infeasible set $B \cup \{v\}$—whose complement is a smaller length-s vertex cut.

Theorem 2. *Algorithm* Minimalize *finds a minimal length-s vertex cut in time $O(n^3)$ and space $O(n^2)$, or, more precisely, in:*

- *$O(nm + n^2 \log\log n + |Q||F_0|)$ time and $O(m + |F_0|)$ space under nonnegative weights;*
- *$O(nm + |Q||F_0|)$ time and $O(m + |F_0|)$ space in the hop-based case.*

Here, F_0 denotes the original set of far pairs, i.e., the set F from step 1.

In Table 1, we compare the performance of formulation CUT with three different settings: (1) without minimalizing the length-s vertex cuts, (2) applying algorithm MinimalizeBasic, and (3) applying algorithm Minimalize. In these tests, we set $s = \text{diam}(G)$ since this is the smallest feasible value of s. Moreover, we exclude the cases where $s = \text{diam}(G) = 2$, as there is nothing to minimalize. Note that these tests use the BestInHeuristic that is described in Section 5. CUT only solves 4 instances if we do not minimalize the length-s vertex cuts, in which case the formulation is practically useless. Using MinimalizeBasic, we can solve 20 of the 30 instances, but the callback time is large in many cases. For example, see the instance v200_d20 in which 37 minutes is spent in the callback. If Minimalize is used instead, the callback time reduces to 1 minute. Moreover, the time saved by Minimalize allows the solver to improve the lower bound in 5 of the 10 unsolved cases.

Table 1: An evaluation of the usefulness of MINIMALIZE as compared to no minimalizing and to MINIMALIZEBASIC. For all graphs G, we set $s = \text{diam}(G)$ and exclude instances where $s = 2$. We report the optimal objective (or the best lower/upper bounds $[L, U]$ after one hour) under the columns labeled obj. We also give the time spent in the callback (call), and the total solve time (total), where a dash indicates > 3600 seconds.

		No minimalizing			MINIMALIZEBASIC			MINIMALIZE		
graph	s	obj	call	total	obj	call	total	obj	call	total
v30_d10	8	15	10.29	280.24	15	0.00	0.01	15	0.00	0.02
v30_d20	5	8	0.01	0.05	8	0.02	0.06	8	0.00	0.03
v30_d30	3	[7,8]	229.27	-	8	0.11	0.21	8	0.02	0.10
v50_d5	14	[26,32]	49.88	-	32	0.02	0.07	32	0.02	0.07
v50_d10	5	[13,20]	31.27	-	18	0.44	0.58	18	0.08	0.18
v50_d20	3	[8,14]	51.24	-	14	0.16	0.25	14	0.03	0.11
v50_d30	3	[6,8]	45.67	-	8	1.69	2.70	8	0.16	1.12
v70_d5	8	[24,36]	14.41	-	32	1.13	1.36	32	0.32	0.53
v70_d10	4	[14,31]	19.93	-	29	3.14	5.77	29	0.44	2.98
v70_d20	3	[8,18]	19.60	-	17	47.27	335.22	17	3.40	305.96
v70_d30	3	[6,8]	66.95	-	7	9.68	12.93	7	0.50	3.33
v100_d5	5	[24,57]	9.71	-	56	16.38	33.14	56	2.10	17.91
v100_d10	4	[14,31]	10.82	-	[22,26]	322.14	-	[22,26]	18.79	-
v100_d20	3	[8,20]	15.26	-	[14,20]	507.29	-	[14,20]	24.16	-
v120_d5	6	[25,40]	8.80	-	31	464.72	1511.87	31	25.86	1087.17
v120_d10	3	[13,68]	7.92	-	63	27.31	35.60	63	2.21	10.31
v120_d20	3	[8,21]	13.89	-	[10,21]	973.37	-	[10,21]	42.57	-
v120_d30	3	[6,12]	19.98	-	[7,12]	737.91	-	[7,12]	21.65	-
v150_d5	5	[25,54]	7.19	-	[29,54]	1088.47	-	[30,54]	121.73	-
v150_d10	3	[13,65]	7.35	-	[41,63]	717.64	-	[43,61]	39.95	-
v150_d20	3	[8,22]	12.70	-	[9,22]	1915.12	-	[9,22]	58.24	-
v200_d5	4	[25,92]	6.10	-	[48,92]	1851.87	-	[49,92]	169.57	-
v200_d10	3	[13,64]	6.26	-	[23,64]	1734.04	-	[26,64]	132.19	-
v200_d20	3	[7,22]	7.58	-	[7,22]	2226.01	-	[8,22]	58.71	-
IEEE-14	5	5	0.00	0.00	5	0.00	0.00	5	0.00	0.01
IEEE-30	6	14	0.33	3.98	14	0.00	0.01	14	0.00	0.01
IEEE-57	12	[24,35]	23.24	-	35	0.02	0.05	35	0.01	0.04
RTS-96	13	[27,40]	13.53	-	37	0.09	0.17	37	0.07	0.14
IEEE-118	14	[39,48]	15.22	-	48	0.01	0.17	48	0.00	0.15
IEEE-300	24	[104,139]	16.49	-	135	14.35	18.00	135	8.65	11.98

4 Fractional Separation

We followed a suggestion of one of the reviewers to use fractional separation. Specifically, we implemented the max-flow/min-cut-based procedure for $s = 3$, as described in the paper, and used it as part of a branch-and-cut approach.

When experimenting with fractional separation, we felt it was important to try different implementations, as the number of cuts and the cut violation can greatly impact the performance. With this in mind, we tried nine different implementations.

The generic pseudocode that we use in the callback is as follows, where x^* is the branch-and-bound node's LP solution, $\epsilon \in \{0.01, 0.1, 0.5\}$ is the cut violation threshold, and $I \in \{1, 2, 3\}$ controls how many cuts are added per callback.

- for every vertex $a \in \{1, 2, \ldots, n\}$ do
  - for every vertex $b \in \{1, 2, \ldots, n\}$ do
    - compute a minimum-weight length-3 a,b-separator $C \subseteq V \setminus \{a, b\}$ in graph $G = (V, E)$, where each vertex i has weight x^*_i;
    - if the cut violation $1 - x^*(C)$ is more than ϵ,
      - add the cut $x(C) \geq 1$;
      - if $I = 1$, then return;
      - if $I = 2$, then break the for-loop over b (i.e., go to the next a);

As can be observed in the pseudocode, setting $I = 1$ adds ≤ 1 cut per callback, setting $I = 2$ adds $\leq n$ cuts per callback, and setting $I = 3$ adds $\leq n^2$ cuts per callback.

There are several ways to speed up separation in practice. In all of our implementations, we use the following speedups:

1. We can skip the cases $a = b$.
2. Our test instances are bidirected, so it suffices to consider $b \in \{a + 1, a + 2, \ldots, n\}$.
3. If $x^*(N^+(a) \cap N^-(b)) + \epsilon \geq 1$, then no sufficiently violated cut will exist, and one can skip to the next vertex b.

We tried each of the three settings $I \in \{1, 2, 3\}$ across three different values of $\epsilon \in \{0.01, 0.1, 0.5\}$, resulting in a total of nine implementations. The results are provided in Tables 2, 3, 4. In each table, we report the lower and upper bounds on the objective value after a one-hour time limit. For reference, we also report the bounds obtained using our integer separation routine that we have been using all along.

Among the nine fractional separation implementations, there is no clear winner. For the graph v150_d10, the best bounds are achieved with the parameter settings $(I, \epsilon) \in \{(1, 0.01), (3, 0.01), (3, 0.1)\}$. While, for the graph v200_d20, the best bounds are achieved with a different parameter setting $(I, \epsilon) = (1, 0.1)$. However, if we include our original implementation into the mix, it is the clear winner. This is despite a good-faith effort towards using fractional separation, including a polynomial-time separation routine for $s = 3$ and helpful speedups. We imagine that the results would be even worse if we had to perform exact separation for $s \geq 5$, where separation is NP-hard. This confirms our suspicions that we had obtained when studying another semi-related problem (Buchanan and Salemi, 2018).

One explanation for the poor performance of $s = 3$ fractional separation is the time to solve the separation problem. It requires the solution of up to $\Theta(n^2)$ different minimum cut problems,

making the runtime something like mn^3. In contrast, our integer separation routine requires $O(n^3)$ time. Note that for the $s = 3$ instance v200_d20, $n^3 = 8,000,000$, while $mn^3 = 31,840,000,000$. Any time spent on the time mn^3 procedure in the callback is time not spent on other things (like branching). Apparently, this tradeoff is not worth it.

As a final remark, see that several of our instances remain unsolved even when $s = 2$. Recall that we add all length-2 vertex cut inequalities upfront in the $s = 2$ case since there are only $O(n^2)$ of them, in which case there is no need to perform fractional separation. Thus, even when fractional separation takes "zero time", the instances cannot be solved. This lends credence to the idea that there is little room for fractional separation to help; these instances may just be challenging. We note that one of our $s = 2$ instances was submitted to MIPLIB 2017 and has been included in their Collection Set. At the time of writing, it is still considered an "open" instance, meaning that no MIP solver has been able to solve it (even with the latest software releases). The instance (and its "open" status) can be found on the MIPLIB website: `http://miplib.zib.de/instance_details_v150d30-2hopcds.html`

Table 2: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.01$

graph	s	Lazy	$I = 1$ ≤ 1	$I = 2$ $\leq n$	$I = 3$ All Pairs
v100_d20	3	[14,20]	[11,20]	[10,20]	[11,20]
v120_d20	3	[10,21]	[8,21]	[7,21]	[8,21]
v120_d30	3	[7,12]	[4,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[38,65]	[35,65]	[38,65]
v150_d20	3	[9,22]	[7,22]	[7,22]	[7,22]
v200_d10	3	[26,64]	[27,64]	[24,64]	[27,64]
v200_d20	3	[8,22]	[6,22]	[6,22]	[6,22]

Table 3: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.1$

graph	s	Lazy	$I = 1$ ≤ 1	$I = 2$ $\leq n$	$I = 3$ All Pairs
v100_d20	3	[14,20]	[10,20]	[10,20]	[11,20]
v120_d20	3	[10,21]	[9,21]	[7,21]	[8,21]
v120_d30	3	[7,12]	[6,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[28,65]	[37,65]	[38,65]
v150_d20	3	[9,22]	[8,22]	[7,22]	[7,22]
v200_d10	3	[26,64]	[19,64]	[24,64]	[26,64]
v200_d20	3	[8,22]	[7,22]	[6,22]	[6,22]

Table 4: Lower and upper bounds after one hour when cut violation threshold is $\epsilon = 0.5$

graph	s	Lazy	$I = 1$ ≤ 1	$I = 2$ $\leq n$	$I = 3$ All Pairs
v100_d20	3	[14,20]	[11,20]	[9,20]	[10,20]
v120_d20	3	[10,21]	[8,21]	[7,21]	[7,21]
v120_d30	3	[7,12]	[4,12]	[4,12]	[4,12]
v150_d10	3	[43,61]	[28,65]	[37,65]	[36,65]
v150_d20	3	[9,22]	[7,22]	[6,22]	[6,22]
v200_d10	3	[26,64]	[20,64]	[23,64]	[24,64]
v200_d20	3	[8,22]	[6,22]	[6,22]	[6,22]

5 The Importance of Providing a Heuristic Solution to the Solver

We provide a simple "best-in" heuristic for the minimum latency-s CDS problem. In the pseudocode, the vertex subset D represents a partial solution; the set F represents the set of "far pairs," i.e., those pairs of vertices that inhibit D from being a feasible solution; and the score of a vertex is how many far pairs it would "close" or eliminate. Note that the scores can increase during the heuristic, and they can all be zero while F is nonempty.

BestInHeuristic(G, s):

1. compute $\text{diam}(G)$;
2. if $\text{diam}(G) > s$, return "infeasible";
3. initialize $D \leftarrow \emptyset$ and $F \leftarrow \{(i,j) \in V \times V \mid (i,j) \notin E,\ i \neq j\}$;
4. while $F \neq \emptyset$ do
   (a) for $v \in V \setminus D$ do
   - compute shortest paths from v in graph $\overrightarrow{G}_v^D$;
   - compute shortest paths to v in graph $\overleftarrow{G}_v^D$;
   - compute $\text{score}(v) := \left|\left\{(a,b) \in F \;\middle|\; \text{dist}_{\overleftarrow{G}_v^D}(a,v) + \text{dist}_{\overrightarrow{G}_v^D}(v,b) \leq s\right\}\right|$;

   (b) let $v^* \in V \setminus D$ be a vertex of maximum score;
   (c) $R \leftarrow \left\{(a,b) \in F \;\middle|\; \text{dist}_{\overleftarrow{G}_{v^*}^D}(a,v^*) + \text{dist}_{\overrightarrow{G}_{v^*}^D}(v^*,b) \leq s\right\}$;
   (d) update $D \leftarrow D \cup \{v^*\}$ and $F \leftarrow F \setminus R$;
5. return D.

This heuristic returns a feasible solution (when the instance is feasible), and its runtime is $O(n^4)$, or, more precisely, $O(n^3 + |D|mn + |D||F_0|n)$, where D is the heuristic solution at the end, and F_0 is the initial set of far pairs from step 3. Given that it takes time $O(n^3)$ to verify feasibility, this heuristic is not too costly. The output of this heuristic is not necessarily inclusion-minimal, so we run a post-processing procedure to make it so.

We also experimented with a "worst-out" heuristic that starts with V as an initial solution and greedily deletes vertices v, in order of decreasing indegree $|N^-(v)|$ plus outdegree $|N^+(v)|$, as long as this is still feasible. This heuristic, when applied to the minimum CDS problem, is "provably best" in the sense of Kahruman-Anderoglu et al. (2016) meaning that, unless P=NP, no polynomial-time algorithm always finds a better solution than this heuristic (when a better solution exists). Similar heuristics work well in practice for the minimum CDS problem (Butenko et al., 2004). However, this particular worst-out heuristic performed poorly compared to the best-in heuristic in initial experiments, so it is excluded.

In Table 5, we evaluate the usefulness of the proposed best-in heuristic (while employing MINIMALIZE to strengthen the inequalities). In most cases, the impact of the heuristic is marginal (positively or negatively). Of the instances that were solved to optimality, its effect is most pronounced on v70_d20 and v120_d5 where it slowed down the solver by 173 seconds and sped up the solver by 409 seconds, respectively. It may seem strange that providing an MIP start could worsen the performance, but this is likely a natural consequence of solver variability, and we did not attempt to tame or take advantage of this behavior. Note, however, that Gurobi was unable to find a feasible solution on four instances if left unaided. Further, on the instances v150_d20 and v200_d20 the best-in heuristic found solutions (in under a second) that were 12 vertices and 13 vertices better, respectively, than what Gurobi found in an hour. Frankly, Gurobi's performance

on some of the $s = 2$ instances surprised us. For example, on the instance v120_d50, CUT has only 120 binary variables and 3570 covering constraints, and yet it did not solve within an hour.

Table 5: An evaluation of the usefulness of BestInHeuristic as compared to running Gurobi without an MIP start. We report the objective of the heuristic solution (hobj) and the time spent in the heuristic (htime). For the meanings of other reported quantities, refer to Table 1.

		No heuristic		with BestInHeuristic			
graph	s	obj	total	hobj	obj	htime	total
v30_d10	8	15	0.01	15	15	0.00	0.02
v30_d20	5	8	0.09	8	8	0.00	0.03
v30_d30	3	8	0.16	8	8	0.00	0.10
v30_d50	2	7	0.02	7	7	0.00	0.01
v30_d70	2	3	0.02	4	3	0.00	0.04
v50_d5	14	32	0.05	32	32	0.01	0.07
v50_d10	5	18	0.18	20	18	0.01	0.18
v50_d20	3	14	0.14	14	14	0.00	0.11
v50_d30	3	8	7.23	8	8	0.00	1.12
v50_d50	2	9	0.18	11	9	0.00	0.17
v50_d70	2	4	0.89	4	4	0.00	0.79
v70_d5	8	32	0.44	36	32	0.01	0.53
v70_d10	4	29	1.34	31	29	0.01	2.98
v70_d20	3	17	132.69	18	17	0.01	305.96
v70_d30	3	7	1.95	8	7	0.00	3.33
v70_d50	2	10	0.76	10	10	0.00	0.56
v70_d70	2	5	2.04	5	5	0.00	1.57
v100_d5	5	56	17.02	57	56	0.04	17.91
v100_d10	4	[22,26]	-	31	[22,26]	0.02	-
v100_d20	3	[14,18]	-	20	[14,20]	0.01	-
v100_d30	2	39	5.35	42	39	0.03	5.00
v100_d50	2	12	59.15	13	12	0.01	61.27
v100_d70	2	5	8.91	5	5	0.00	8.17
v120_d5	6	31	1496.49	40	31	0.05	1087.17
v120_d10	3	63	10.33	68	63	0.06	10.31
v120_d20	3	[10,26]	-	21	[10,21]	0.02	-
v120_d30	3	[7,12]	-	12	[7,12]	0.01	-
v120_d50	2	[11,12]	-	13	[11,12]	0.01	-
v120_d70	2	5	32.86	6	5	0.00	29.67
v150_d5	5	[30,∞]	-	54	[30,54]	0.10	-
v150_d10	3	[39,∞]	-	65	[43,61]	0.10	-
v150_d20	3	[9,34]	-	22	[9,22]	0.04	-
v150_d30	2	[35,42]	-	45	[35,41]	0.06	-
v150_d50	2	[9,13]	-	13	[9,13]	0.02	-
v150_d70	2	6	560.68	7	6	0.01	569.10
v200_d5	4	[46,∞]	-	92	[49,92]	0.34	-
v200_d10	3	[23,∞]	-	64	[26,64]	0.20	-
v200_d20	3	[7,35]	-	22	[8,22]	0.07	-
v200_d30	2	[27,43]	-	45	[27,44]	0.12	-
v200_d50	2	[8,15]	-	16	[8,15]	0.04	-
v200_d70	2	[4,6]	-	7	[4,7]	0.01	-
IEEE-14	5	5	0.00	5	5	0.00	0.01
IEEE-30	6	14	0.01	14	14	0.00	0.01
IEEE-57	12	35	0.02	35	35	0.01	0.04
RTS-96	13	37	0.15	40	37	0.02	0.14
IEEE-118	14	48	0.07	48	48	0.06	0.15
IEEE-300	24	135	8.91	139	135	2.31	11.98

6 Results for the Fault-Tolerant Variant

In Table 6, we provide results for the fault-tolerant variant, where $r = 2$, and s is set to the smallest feasible value[1] $\max\{\text{diam}(G - v) \mid v \in V\}$. The reason for including results for $r = 2$ (and not for larger values) is that network topologies that continue to function after the failure of a single entity are often considered sufficient for practical purposes (Monma and Shallcross, 1989; Grötschel et al., 1992).

The implementation needed to be modified to accommodate $r = 2$. For one, we needed to alter the heuristic. Our approach is to first find a feasible solution $D \subseteq V$ for $r = 1$ using the BestInHeuristic from before. Then we consider some $v \in D$ and check if $D \setminus \{v\}$ is a latency-s CDS. If not, then we augment it by adding high *score* vertices from $V \setminus D$ in the same way as in BestInHeuristic until the failure of $v \in D$ will not cause problems. We do this for each vertex $v \in D$ from the initial solution (for $r = 1$) until our heuristic solution is feasible for $r = 2$.

We also had to modify the callback routines. When the solver encounters a vertex subset D that satisfies the initial constraints, we check if it is a latency-s CDS (for $r = 1$). If not then $V \setminus D$ is a length-s vertex cut, and we strengthen it using Minimalize as before. Supposing D is a latency-s CDS, then we check if it is a solution for $r = 2$, i.e., if $D \setminus \{v\}$ is a latency-s CDS for each $v \in D$. If not, then $(V \setminus D) \cup \{v\}$ is a length-s vertex cut, and we use Minimalize on it.

References

Austin Buchanan and Hosseinali Salemi. Parsimonious formulations for low-diameter clusters, 2018. `http://www.optimization-online.org/DB_HTML/2017/09/6196.html`.

Sergiy Butenko, Xiuzhen Cheng, Carlos A Oliveira, and Panos M Pardalos. A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks. In *Recent developments in cooperative control and optimization*, pages 61–73. Springer, 2004.

Martin Grötschel, Clyde L Monma, and Mechthild Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.

Sera Kahruman-Anderoglu, Austin Buchanan, Sergiy Butenko, and Oleg A. Prokopyev. On provably best construction heuristics for hard combinatorial optimization problems. *Networks*, 67(3): 238–245, 2016.

Daniel Lokshtanov, Dániel Marx, Saket Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105), 2013.

Clyde L Monma and David F Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, 1989.

Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524. ACM, 2013.

[1] If the graph has a cut vertex, we exclude it from consideration since there is no feasible (finite) value for s.

Table 6: Results for the fault-tolerant variant, where $r = 2$ and $s = \max\{\mathrm{diam}(G - v) \mid v \in V\}$. Only biconnected graphs are considered. The number of branch-and-bound nodes is "BB node."

graph	diam	s	BB node	htime	hobj	obj	total
v30_d30	3	3	462	0.00	12	12	0.09
v30_d50	2	2	0	0.01	15	14	0.02
v30_d70	2	2	0	0.00	6	5	0.02
v50_d10	5	5	239	0.12	34	32	0.33
v50_d20	3	4	432	0.03	15	13	0.25
v50_d30	3	3	10625	0.01	13	12	2.28
v50_d50	2	2	9	0.02	18	15	0.19
v50_d70	2	2	0	0.00	7	6	0.33
v70_d5	8	10	201	0.27	53	51	0.61
v70_d10	4	5	7519	0.13	33	27	4.44
v70_d20	3	3	384357	0.12	31	26	106.28
v70_d30	3	3	29771	0.02	14	12	7.30
v70_d50	2	2	174	0.03	18	16	0.56
v70_d70	2	2	163	0.01	9	7	1.77
v100_d5	5	6	134933	0.71	63	56	102.40
v100_d10	4	4	943671	0.29	48	[38,41]	-
v100_d20	3	3	1260796	0.15	32	[24,27]	-
v100_d30	2	3	366610	0.07	18	[11,16]	-
v100_d50	2	2	21597	0.07	19	18	36.17
v100_d70	2	2	4146	0.02	10	8	14.25
v120_d5	6	7	134501	1.22	59	50	164.92
v120_d10	3	4	570958	0.56	42	[28,39]	-
v120_d20	3	3	357951	0.28	32	[18,30]	-
v120_d30	3	3	265467	0.08	19	[10,17]	-
v120_d50	2	2	1109463	0.13	22	[16,18]	-
v120_d70	2	2	6075	0.02	10	8	37.63
v150_d5	5	5	388468	2.44	80	[53,80]	-
v150_d10	3	4	283107	0.49	45	[25,45]	-
v150_d20	3	3	273843	0.51	36	[16,34]	-
v150_d30	2	3	244025	0.20	19	[9,19]	-
v150_d50	2	2	222265	0.18	22	[15,18]	-
v150_d70	2	2	628745	0.04	11	[8,9]	-
v200_d5	4	4	644105	12.81	128	[104,118]	-
v200_d10	3	3	426782	5.56	95	[55,95]	-
v200_d20	3	3	224602	0.79	32	[13,32]	-
v200_d30	2	2	77784	6.57	72	[49,63]	-
v200_d50	2	2	33061	0.64	25	[13,22]	-
v200_d70	2	2	119071	0.07	12	[7,9]	-